
# Evaluating the Navigation in Software Visualization in Augmented Reality with Speech Processing

BACHELORARBEIT

für die Prüfung zum

**Bachelor of Engineering**

im Studiengang Informationstechnik

an der Dualen Hochschule Baden-Württemberg Mannheim

von

Adrian Stock

| | |
|---|---|
| Bearbeitungszeitraum | 24.06.2019 bis 15.09.2019 |
| Kurs | TINF16ITIN |
| Matrikelnummer | 4808954 |
| Ausbildungsfirma | Deutsches Zentrum für Luft- und Raumfahrt e.V. Köln |
| Gutachter der Ausbildungsfirma | Prof. Dr. Nico Hochgeschwender |
| Gutachter der Studienakademie | Prof. Dr. Harald Kornmayer |

# Ehrenwörtliche Erklärung

Gemäß §5 (3) der „Studien- und Prüfungsordnung DHBW Technik“ vom 22. September 2011.
Ich habe die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet.

Mannheim, den 11. September 2019

# Abstract

Architecture of software systems is complex and abstract. Visualizing it is one approach to improve comprehension of software architecture. The recent developments in the area of virtual reality and augmented reality allow software to be visualized in a three-dimensional environment.

This representation of software architecture enables users to navigate through the different elements of an application. In Augmented Reality, gestures are used to explore the components of the software and their dependencies. However, the number of gestures is limited to only a few. Therefore, a speech assistant was developed to provide users more functionalities when exploring software systems.

This thesis examines the navigation in an application for software visualization in Augmented Reality using speech processing. A comparative user study, which compares the navigation with speech to gestures, is carried out. This study evaluates whether speech processing enhances the usability of software visualization in augmented reality.

The objective of the present work is to discuss both the theoretical and technical aspects when setting up the study. Different definitions of usability are considered which serves as a foundation for the study. Furthermore, the developed speech assistant is integrated into an application for software visualization.

Based on the results of the user study, the usability of speech processing for exploring the architecture of software systems is evaluated.

# Kurzfassung

Die Architektur von Software ist komplex und abstrakt. Sie zu visualisieren ist eine Möglichkeit um das Verständnis der Softwarearchitektur zu verbessern. Aktuelle Entwicklungen im Bereich der Virtual Reality und Augmented Reality ermöglichen es Software im dreidimensionalen Raum dargestellt zu werden.
Durch die Darstellung der Softwarearchitektur können Benutzer durch die unterschiedlichen Elemente einer Applikation navigieren. Gesten werden in Augmented Reality verwendet, um die unterschiedlichen Komponenten einer Software und deren Abhängigkeiten zu erforschen. Die Anzahl der Gesten ist jedoch auf einige wenige beschränkt. Um bei der Erkundung von Software mehr Funktionalitäten bereitzustellen wurde ein Sprachassistent entwickelt.
Die vorliegende Thesis untersucht die Navigation mit Sprachsteuerung in einer Applikation zur Softwarevisualisierung in Augmented Reality. Dazu wird eine vergleichende Nutzerstudie durchgeführt. Diese vergleicht die Navigation mit Sprach- und Gestensteuerung. Die Studie evaluiert, ob Sprachverarbeitung die Usability von Softwarevisualisierung in Augmented Reality verbessert.
Diese Arbeit untersucht sowohl den theoretischen als auch den technischen Aspekt beim Erstellen der Studie. Unterschiedliche Definitionen von Usability werden verglichen und als Grundlage der Nutzerstudie verwendet. Des Weiteren wird der entwickelte Sprachassistent in eine Applikation zur Softwarevisualisierung eingebunden.
Basierend auf den Ergebnissen der Nutzerstudie wird die Usability der Sprachverarbeitung zur Erkundung der Architektur von Software evaluiert.

# Contents

# List of Figures

# List of Tables

# Listings

# List of Abbreviations

| | |
|---|---|
| **AR** | Augmented Reality |
| **CI** | Conversational Interface |
| **DLR** | German Aerospace Center (Deutsches Zentrum für Luft- und Raumfahrt) |
| **GUI** | Graphical User Interface |
| **IVS** | Intelligent and Distributed Systems |
| **LoC** | Lines of code |
| **MVC** | Model-View-Controller |
| **NLG** | Natural Language Generation |
| **NLU** | Natural Language Understanding |
| **OSGi** | Open Service Gateway initiative |
| **PDA** | Personal Digital Assistant |
| **RCE** | Remote Component Environment |
| **REST** | Representational State Transfer |
| **UML** | Unified Modeling Language |

| | |
|---|---|
| **VR** | Virtual Reality |
| **XML** | Extensible Markup Language |

# 1. Introduction

Over time, growing software projects become more confusing and complex [1]. Due to the interaction of many classes, the overview over the different dependencies is quickly lost. Especially people who have not been involved in the development process so far are severely affected [1]. This makes maintaining sustainable software, which displays one of the most costly aspects in software engineering [2], more difficult.
In order to counteract these problems the usage of software visualization is proposed. It allows software to be visualized at a higher level of abstraction [3]. Furthermore, it strengthens the understanding of the software. Hence, different attempts have already been made to visualize software [4]. The software presented in this thesis, IslandViz, uses the *island metaphor* for visualizing the architecture of a software system [5].
IslandViz (island visualization) was developed in the Department of Intelligent and Distributed Systems of the Institute of Simulation and Software Technology at DLR. It serves to visualize software based on OSGi in a three-dimensional space. OSGi is a framework for Java, which adds the concepts of bundles and services to the already existing Java functions to increase the modularization of the code. OSGi projects also make use of XML files which contain meta information of the software. Common methods, such as UML diagrams, are hardly sufficient for visualizing these relationships. Therefore, the approach of visualization in 3-dimensional space was chosen. In 2017 the development of IslandViz started and was implemented for virtual reality (VR) [5]. One year later, in 2018, the application was ported to another medium, augmented reality (AR) [6]. This work focuses on the implementation of IslandViz in AR.

## 1.1. Motivation

In [7], a user study was carried out to evaluate the usability of IslandViz in comparison to a two-dimensional software visualization application. The study shows that some simple tasks in IslandViz are cumbersome to perform. Some of these tasks, however, are crucial to exploring a new software. This results in the necessity of providing new options to enhance exploring in IslandViz. In AR the user executes actions by using different gestures [8], wherease in VR the same actions can be performed with controllers. AR limits the number of possible actions through the small number of available gestures, though. Thus, in order to give the application more functionalities, a concept was developed to extend IslandViz with a conversational interface [9]. This conversational interface can be seen as a speech assistant which helps the user to explore the software in IslandViz. The user uses voice commands to operate the conversational interface. The conversational interface uses a service developed in prior work[10]. The so-called NLU service receives a written phrase, which represents the voice command entered by the user, and interprets which action the user wants to perform. Before the development of this conversational interface is further pursued, it is to be checked how the addition of such a feature impacts the usability of IslandViz. This leads to the following research question: *Does a conversational interface improve the usability of IslandViz?* In order to examine this question, a user study is carried out. On the basis of the results, the usability of the conversational interface is evaluated and the research question answered.

## 1.2. Structure of the Thesis

**Chapter 2** discusses the application IslandViz and the OSGi-based software RCE is briefly introduced. The latter is used in this thesis as an example to be visualized by IslandViz. Furthermore, the extension of the software architecture of IslandViz by a conversational interface is presented.

**Chapter 3** introduces the concept of usability and derives hypotheses, which are examined and used to answer the actual research question.

**Chapter 4** focuses on the process of creating a user study. This study is designed in such a way that its results are used to confirm or deny the hypotheses previously created. Since the conversational interface is still under development, IslandViz in its current form cannot be used for the user study yet. In order to conduct the study in a representative manner, some changes to the application were necessary.

**Chapter 5** therefore implements new code in order to be able to use the functions of voice control in the program. Afterwards, the user study is carried out.

**Chapter 6** then discusses the results of the study and evaluates the different hypotheses on this basis. Subsequently, the actual research question is examined.

**Chapter 7** evaluates and summarizes the results. Furthermore, recommendations are given on how to continue the development of IslandViz.

# 2. Technical Background

IslandViz serves to visualize software that is based on OSGi in a three-dimensional environment. The application uses different elements as metaphors to portray the components of an OSGi-based software. In previous work, a concept was discussed to include a speech processing service, the conversational interface, to IslandViz [9].

## 2.1. The Framework OSGi

OSGi (Open Services Gateway initiative) is a framework that supports the implementation of component-based, service-oriented applications in Java [11]. It addresses two problems which are inherent in the modular system in Java:

- The information hiding principle is applied to the level of classes. Hence, every public class can be accessed from every other class. As a consequence, a system can easily end up highly coupled.
- The modular system is static. Modules cannot be updated during runtime. The application has to be restarted to add, remove, or change a module.

In 1998, the OSGi Alliance has proposed the OSGi framework[11]. It provides a dynamic, component-based, service-oriented modular system for Java. OSGi systems are built around independent modules, which provide well-defined services. These modules are called bundles. The life cycle of those bundles is specified by the OSGi standard within a runtime infrastructure. This allows developers to dynamically add and remove

them during runtime.
A bundle is divided into several packages which are divided into several classes. To use a class as a service, it needs to be defined in an XML file as such. The manifest, which is also stored in the bundle, is used to declare static information about the bundle, such as the packages it imports and exports. Exporting a package allows other bundles to import it and therefore access the services within this package.

## 2.2. Visualizing OSGi-based Software

The software system visualized by IslandViz is represented as an ocean filled with islands. Fig. 2.1 shows an overview over many different islands. However, in this

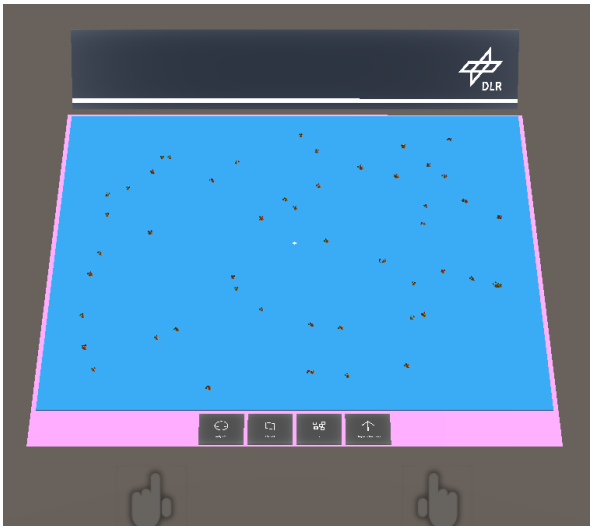
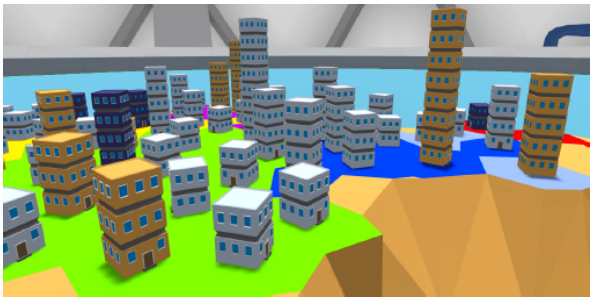



Figure 2.1.: An overview over IslandViz shows the ocean and many different islands

perspective, the islands are hardly recognizable. The images in Fig. 2.2 show more detailed views.
Each island represents a different bundle. An island is divided into several regions which each visualize a package within this bundle. Fig. 2.2a[1] shows a more detailed

[1]taken from [5], p.33

(a) An island with different regions



(b) Close up view of the buildings

Figure 2.2.: Different views on an island

view of an island. The different colours on the ground visualize the packages. Fig. 2.2b[2] shows a close-up look of the different buildings on an island. Each building represents an individual class type, e.g. classes or services, and is placed in on of the regions. All buildings standing on the same coloured region belong to the same package. The size of an island is proportional to the number of buildings on it. The height of a building indicates the number of lines of code of the belonging class.

Two ports, a red one and a green one, are added to each island. The green port represents the imports of the corresponding bundle, while the red one shows the exports. Fig. 2.3[3] shows several islands and a specific dependency, depicted by the arrow.

The different metaphors described in this section are summed up in Table 2.1.

[2]taken from [5], p.34

[3]taken from [12]

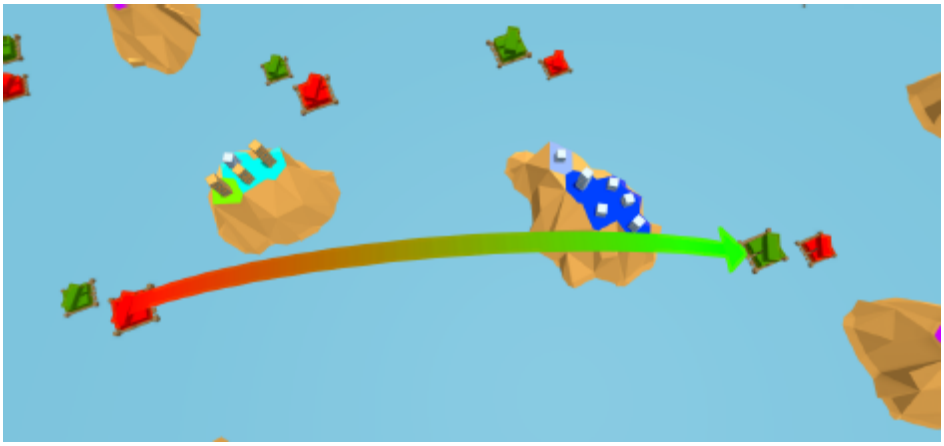


Figure 2.3.: Visualization of a dependency between two packages

| **Component** | **Visualization** |
|---|---|
| Software system | Ocean |
| Bundle | Island |
| Package | Region |
| Individual class type | Buildings |
| Number of lines of code | Height of a building |
| Dependencies | Ports |

Table 2.1.: Summary of the different metaphors

## Considering Previous Usability Evaluations

The interactions with the environment are limited to gestures. More complex tasks cannot be performed easily. In [7] a usability evaluation has shown that simple tasks are cumbersome to perform. In the evaluation, seven test users had to perform five tasks. Table 2.2 shows two of them and the average time the users needed to finish the task. These tasks have been performed using the VR application. However, the problems portrayed in the evaluation in [7] apply to the AR implementation as well.

| **Task** | **Mean time in seconds** | **Error rate in percent** |
|---|---|---|
| Select the bundle with the largest amount of packages. | 55.43 | 14.28 |
| Select the bundle "RCE Core Utils Scripting". | 288.43 | 14.28 |

Table 2.2.: Mean time to finish different tasks in the VR IslandViz

Especially the process of selecting a specific bundle or any other unit is meant to be a common use case and, therefore, must be easy to perform. However, the mean time for this task is about five minutes. Due to the limitations of the controllers and gestures, the user has to visit each island until he finds the desired unit. This is an elaborate process, especially compared to the simplicity of the task. The same goes for the task *Select the bundle with the largest amount of packages*. In this case, the mean time is a lot lower, as the user can make use of the provided visual help, as the size of an island scales with the number of belonging bundles. However, a mean time of one minute is still a lot of time, compared to the task's simplicity. To enhance the user experience and lower the amount of time for simple tasks, IslandViz was extended by a conversational interface.

## Extending the Architecture of IslandViz

IslandViz was extended with a conversational interface as a new modality to enhance its usability. In order to add this functionality, the former software architecture, consisting of just a device to display the visualization and a repository, which holds the data that is to be visualized, was exchanged by a MVC (Model-View-Controller) based approach. The so-called reference architecture is shown in Fig. 2.4[4]. The proposed architecture

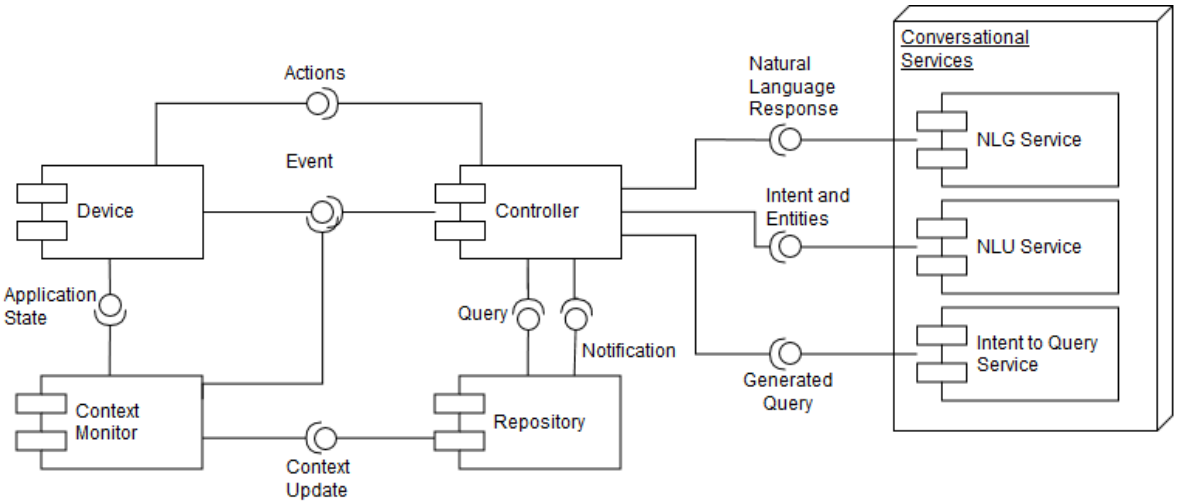



Figure 2.4.: Diagram of the new architecture of IslandViz

allows to spread different competences among several components. This results in a loose coupling of the different components. The components Repository, Device, and Controller implement the MVC pattern.

To understand how this architecture is applied to IslandViz, the following sections focus on the different components and their purpose. At first, the Repository, Device, and Controller are discussed. Thereafter, the different Conversational Services and their purpose are brought into focus. Finally, the remaining component, Context Monitor is described.

[4]Taken from [9]

## 2.3. Storing the Data of a Software System

The Repository is a database that contains the data about the software that is to be visualized. Every software project based on OSGi can be stored in the Repository. On start, IslandViz parses the data stored in the Repository and builds the visualization accordingly. While the application is running, the Repository can be queried by the Controller. This is important for the conversational interface and is further discussed in Section 2.6.
In this thesis, the software RCE is exemplary visualized by IslandViz. Therefore, data about its structure is stored in the Repository.

### RCE

RCE (Remote Component Environment) is an open source project based on OSGi. This software framework uses a GUI to support the creation and execution of workflows. A workflow consists of several components, which are connected to each other. These components can reach from simple components, that store certain variables in a file, to complex ones, that perform all kinds of simulations. The users are able to integrate components themselves. RCE is mostly used by engineers to manage complex simulations [13].

Fig. 2.5 shows an example of a workflow in RCE. Here, the workflow serves to minimize the mass of an intercontinental winged passenger transport [14]. The different components, portrayed as squares or circles, are connected to each other. Each connection is directed, visualized by an arrow, to show the direction of the data flow.

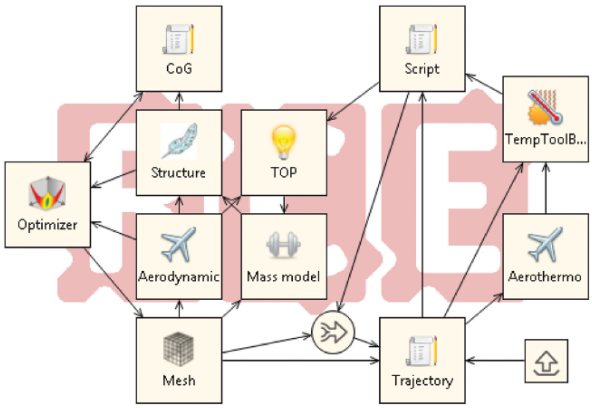



Figure 2.5.: Example of a workflow in RCE

## 2.4. Software Visualization in Mixed Reality

The physical device is the gadget that is used to visualize the application. The Device addressed in the reference architecture (see Fig. 2.4) refers to the software running on this physical device. Its only purpose is to visualize the data stored in the Repository and communicate with the Controller and the Context Monitor. Further responsibilities are outsourced to the Controller and different services. This enables the Device to run smoothly when visualizing the data.

IslandViz is implemented for two different devices. The first development of IslandViz was made for the *HTC Vife*[5], which is a device to simulate a virtual reality. Additionally, IslandViz was ported to the *Microsoft HoloLens* one year later[6]. The HoloLens is a device to display an augmented reality.

In Fig. 2.6 a continuum is displayed, which puts the Real environment on the left side and the Virtual environment on the right sight. Virtual reality is placed on the very right of the continuum, as the user does not see any element of the real environment in VR. AR, however, is placed left from the middle in the continuum. It includes virtual elements within the field of view of a user. However, for the most part, the user sees elements of the real environment.

Fig. 2.6 serves to show, that AR and VR are not completely different from another. Instead, they can both be placed on the same scale on different positions.

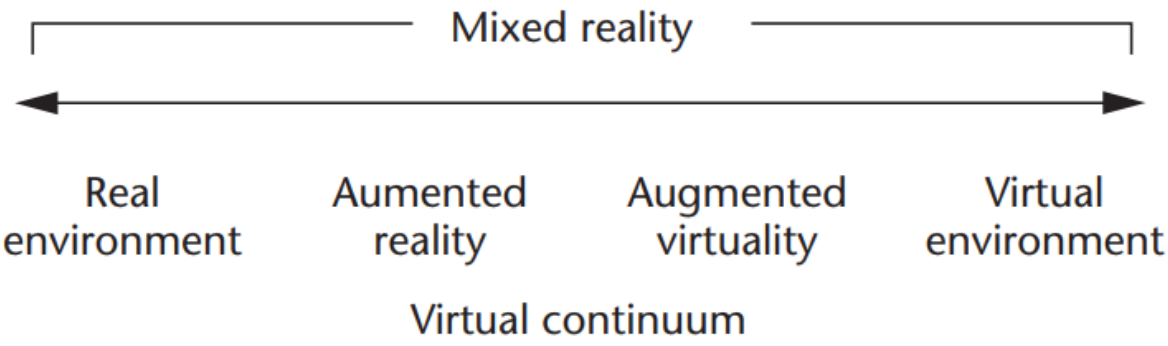



Figure 2.6.: The different visualization methods on the Mixed Reality Continuum

The implementations of IslandViz for VR and AR are presented in the following paragraphs.

## IslandViz in Virtual Reality

The user has two controllers to navigate through the application, zoom in and out, and rotate the visualization. The controllers can also be used to select different units, such as islands, buildings, etc. The left controller can be turned around to see the virtual PDA (Personal Digital Assistant). The PDA shows information about the selected unit. The right controller can also be used to set different options in the PDA. Fig. 2.7

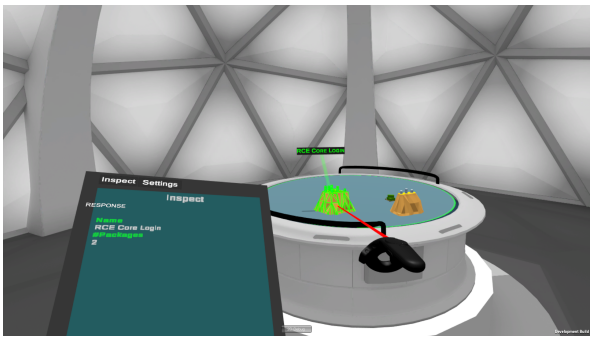



Figure 2.7.: Controller in IslandViz

shows the controllers within the VR environment. The red laser serves to select a unit.

The selected unit is highlighted and related information are shown on the PDA.

## Extending IslandViz to Augmented Reality

The AR implementation does not use any controllers. Instead, both one-handed and two-handed gestures are used to navigate through the application. Furthermore, in the AR implementation, IslandViz uses a state machine. Besides some initial states, the application consists of five states. The gestures have different effects in different states. [8] explains how to perform the different gestures. The next paragraphs focus on the different states and the gestures the user can perform being in this certain state.
In the *state_setup*, the visualization is surrounded by a blue grid as shown in Fig. 2.8. By performing a Tap-and-Hold, the user can position the visualization as desired. To

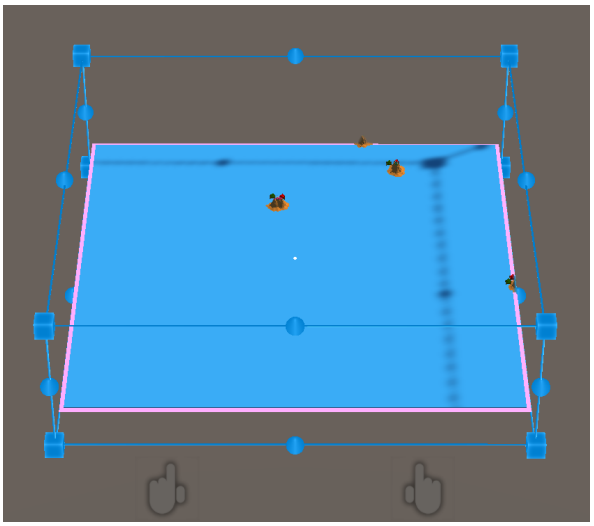


Figure 2.8.: Bounding box to reposition the visualization

confirm the position and go to the next state, the user must perform an Air-Tap.
IslandViz uses a physic engine. Furthermore, it scans the environment and recognizes physical objects in the surrounding area. Due to this, the application takes the environment into account during the repositioning of the visualization. For example, it cannot be dragged inside of a table. However, the visualization can be perfectly placed

on top of it. Therefore, it seems like the ocean, which is a digital object, stands on top of a table, which is a physical object.

After exiting the state_setup, IslandViz gets to the next state, the *state_main*. In this view, the user can navigate freely through the application. In the images shown in Fig. 2.1, IslandViz is in the state_main. By performing a Tap-and-Hold, the user can drag the ocean and move the content of the visualization into every direction. The Two-handed Tap-and-Hold allows the user to zoom in and out. Tapping via the Air-Tap selects the island which is currently focused and gives a close-up view of it. The *focus* is a white dot in the middle of the field of view. Hence, a *focused island* refers to the island which is currently beneath this white dot, respectively the focus. When selecting an island, the application goes into a new state, the *state_inspectIsland*.

state_inspectIsland shows a more detailed view on a selected island as shown in Fig. 2.9a. As explained above, packages, compilation units, imports, and exports of the


(a) Close-up view on an island

(b) Panel with information about the selected bundle

Figure 2.9.: Shown view in the state_inspectIsland

bundle corresponding to the selected island are visualized. Furthermore, the panel in the background enlarges and presents information about the bundle. Fig. 2.9b shows

a close-up view on this panel. It depicts the packages, exports, and imports of the selected bundle.
In this state, the gestures have different effects than when being in the state_setup. The Air-Tap has different effects, depending on the focused object by the user. Performing the Air-Tap while focusing a package or a compilation unit, the user selects either the focused package or the package which contains the focused compilation unit. When the user does an Air-Tap while having the island itself or the ocean focused deselects the island. The application then goes back into the state_setup. Air-Tapping while focusing a harbour shows either the imports or exports of this bundle, depending on which harbour is currently focused. The Tap-and-Hold gesture allows the user to rotate the view around the island, and thus, viewing the island from different angles.

When selecting a package, IslandViz goes into *state_inspectRegion*. The view does not change. However, the selected package is highlighted. Furthermore, the text on the panel changes, as it now presents information about the package. The names of the bundle, package, and the contained compilation units are displayed on the panel. Being in this state, the user can select a compilation unit, select another package within this bundle, show imports or exports, or deselect the package by performing the Air-Tap. The Tap-and-Hold still allows the user to rotate the view around the island.

When selecting a compilation unit, IslandViz changes to the last of the five states considered in this section, *state_inspectBuilding*. Again, the view does not differ from the view of state_inspectIsland or state_inspectRegion and the Tap-and-Hold still allows the user to rotate the view. The Air-Tap gesture can be used to select another class with this package, select another package, select imports or exports, or deselect the class. Furthermore, the selected compilation unit is highlighted and the panel shows information about it. The names of the bundle, package, and compilation unit are displayed on the panel. Furthermore, it shows the number of lines of code, the so-called Access Modifier, which states whether the class is public or private, and the type of the unit. The type can be something like Class, Interface, Enum, etc.

The effect of a gesture while being in a certain state is depicted in Table 2.3. The

| State | Air-Tap | Tap-and-Hold | Two-handed Tap-and-Hold |
|---|---|---|---|
| Setup | Confirm the position | Reposition the visualization by dragging it | - |
| Main | Select a bundle | Navigate over the ocean | Zoom in or out |
| Inspect-Island | Select a package, Show imports / exports, go back to the Project-Level | Rotate the view around the island | - |
| Inspect-Region | Select a class, select another package, go back to the Project-Level | Rotate the view around the island | - |
| Inspect-Building | Select another class of the same package, select another package within the same bundle, go back to the Project-Level | Rotate the view around the island | - |

Table 2.3.: The effect of different gestures while being in different levels

start of this section states, that the Device is solely responsible for visualizing and communicating with the Controller and the Context Monitor. With the addition of the conversational interface, which is explained later in this chapter, Text-to-Speech and Speech-to-Text are mandatory. The Microsoft HoloLens offers the possiblity to connect to such a services, which run extern. These services can only be used by the

HoloLens. Hence, the HoloLens is also responsible for making use of this services. The development has shown that this does not cause any issues for the Device when running IslandViz.
The user study carried out in this thesis focuses on the implementation of IslandViz in AR. Therefore, the term IslandViz is used for the further course of this work as a reference to the implementation of the application in AR.

## 2.5. Adding a Central Coordination Unit to IslandViz

The Controller serves as a central unit which establishes the exchange of data between the Device and the Repository. Furthermore, the integration of different services is simplified by the usage of a Controller. The division of the components in IslandViz in Repository, Device, and Controller ensures that not only the services of the conversational interface, as shown in the reference architecture, but any kind of service can be integrated into IslandViz.

## 2.6. Adding Functionalities with Speech Processing

The proposed conversational interface consists of three different services. After the Device receives a spoken input by the user, it makes use of the Speech-to-Text service and transforms the input into a string. This string is sent to the Controller, which forwards it to the *NLU Service* (Natural Language Understanding).
The NLU Service interprets the sentence and breaks it down to an intent and several entities. The intent states the key message of the utterance and what the user wants to achieve by saying this sentence. The entities give additional information to specify the sentence. The concept of intent and entities is further discussed later in this section.
After receiving the intent and entities from the NLU Service, the Controller can make use of the *Intent to Query Service* to query the Repository for further information to

process the user's utterance, if needed.
Based on the results of the preceding two services, the *NLG Service* (Natural Language Generation) creates an utterance which is given to the user as an answer.

### 2.6.1. Understanding Natural Language

When saying a sentence to a system that processes speech inputs, the user desired to achieve a certain goal. The goal of the utterance *Please zoom in on the application* enlarge the components visualized so that the user is able to examine these components more closely. The sentence can be mapped to an intent *zoom_in*. Hence, the intent displays the basic statement of the user without storing the whole sentence. Furthermore, several expressions can be mapped to the same intent. Sentences like *Give me a closer look on the application* or *Show me a close-up view of the visualization* can be interpreted so that the user wants to zoom into the application. However, in many cases, extracting only the intent is not enough to display the content of a statement. If a user expresses the sentence *Select the component de rcenvironment core component*, the intent of the statement can be summarized as *select_component*. This intent does not represent the whole content of the expression, as it does not state, which component is to be selected. One might argue, that the intent should be extended to *select_component_de_rcenvironment_core_component*. However, this means, that for every bundle, a new intent had to be added to the service. This does not scale well, as for every system displayed by IslandViz, the NLU Service has to be adjusted.
To solve this problems, the NLU Service makes use of entities as well. Entities store the information that are not covered by the intent. In the preceding example, the entity *de rcenvironment core component* is added to the intent *select_component*. This combination of intent and entities cover the most important aspects of an utterance. Each sentence is mapped to exactly one sentence, which is extended by any number of entities.
In [10] a first implementation of the NLU Service was developed. The current version of the NLU Service is able to recognize seven different intents:

- `Zoom_in` / `Zoom_out`: Use the conversational interface to zoom in or out without using the controllers.
- `Select_component`: Ask for a specific entity of the software architecture to be selected.
- `Select_biggest_component` / `Select_smallest_component`: Ask for the biggest / smallest entity within another entity to be selected. For example "show the biggest class in the bundle core gui help".
- `Count_components`: Ask for the number of a specific entity type within another entity. For example "how many classes are in the bundle core gui help".
- `Summarize_information`: Ask the chatbot to summarize the information about an entity.

The intents are predefined. The NLU Service can map sentences only to one of those intents. The entities, however, are not predefined. Through the context provided by utterance, the system is able to recognize the entities on its own.
Currently, the model learns from 2200 different utterances. 70% of these utterances are used for the training itself, while the other 30% are used to test the model. The training results in a model with an accuracy of about 90%, which means that out of ten utterances, nine are classified correctly[10].

### 2.6.2. Database Queries to Access Further Information

The seven intents presented above can be put into categories. The intents *Zoom_in*, *Zoom_out.* and *Select_component* are used to navigate within the application. The remaining four intents can be used to gain information about the application. Especially the intents to select the biggest or smallest component and to count components need to query the database within the Repository to gain access to the desired information. The Intent to Query Service makes use of the data extracted by the NLU Service to

gather such information.
Currently, this service is not implemented. As this thesis focuses on the intents that are used to navigate in IslandViz, the service is not required anyways. However in future development, the Intent to Query Service is crucial to enable the full potential of IslandViz.

### 2.6.3. Manlike Interaction with Natural Language

The NLG service generates a natural sentence in response to a user input. In the above example, such an answer could be *Here is the Island you are looking for.* This sentence could then be given after the application has navigated to the required island. What at first sight seems to be a nice feature to make the interaction with IslandViz more human can also offer additional value to the application. For example, if the NLU service has a low confidence in classifying the sentence, the response sentence could help the user understand what the application has done. For example, if the user says the phrase *show me the biggest island*, the application should navigate to the largest of the displayed islands and output a corresponding response phrase. However, if the NLU service fails and interprets the intent as *select_smallest_component*, a spoken response like *This is the smallest island in the software system* might indicate to the user that the application was not performing the action the user originally wanted.
Similar to the Intent to Query Service, the NLG Service is not implemented yet, but not necessary either to carry out the user study. Even though the NLG service allows a more *real* communication with the device and allows conclusions to be drawn about possible sources of error, should the application fail, it is not absolutely necessary. As the intents are used to navigate within the application, the users can actually see, whether the performed action was according to their intent. Hence, as reaction in form of a speech answer is not crucial in the present work.

## 2.7. Including the Context in Speech Processing

The remaining component presented in Fig. 2.4, the Context Monitor, serves as an addition for the conversational interface. It allows an even more natural and easier handling of IslandViz. By using the Context Monitor, the current context can be taken into account in the user's input. After the user has selected an island, sentences like *select the biggest building in this island* can be understood by IslandViz. The word *this* then refers to the island selected.
Similar to the prior to services, the Context Monitor is not implemented yet, but is not crucial for the user study.

## 2.8. Communication Between Different Components

The different components of the reference architecture are executed on different hardware components. For example, the device can be implemented by the HoloLens, while the controller can be executed on a computer. This means that it must be defined how and with which protocol the components communicate with each other. IslandViz uses Hyper Text Transfer Protocol (HTTP) to establish the communication between the components via a web interface. While the code of the Device and the Controller are written in C and used built-in methods to use HTTP, the NLU service, which is based on Python, uses the microframework Flask. Flask is a lightweight microframework based on Python. It can be used to build RESTful web services [15].
As a consequence of this loose coupling, each component can be exchanged during runtime.

## 2.9. Related Work

IslandViz makes use of islands as a metaphor to visualize software based on OSGi. However, different metaphors are possible for this purpose. Both two-dimensional and three-dimensional approaches are conceivable. In 2011, Caserta and Zendra give an overview over different software visualization systems [1].
Graham et al. proposed to visualize software through a solar system in a two-dimensional environment in 2004 [16]. A virtual galaxy represents the software system. Planets, which symbolize the classes, are in the orbit of a central star, which visualizes the belonging package. In its current form, however, the approach is not suitable to represent software based on OSGi, as the bundles cannot be represented yet.
Due to the recent developments in VR and AR, software visualization in a three-dimensional environment becomes more common. In [17], the *city metaphor* is used in order to visualize software in a virtual, three-dimensional environment. It displays a common approach to represent software in virtual reality. Possible metaphors are shown in Table 2.4 (compare [17]).

This approach can easily be extended, e.g. by grouping different cities and let this

| Visualization Level | Code Element |
|---|---|
| World | Software system |
| Country | Directory structure, e.g. package in Java |
| City | File from the software system |
| District | Class |
| Building | Method |

Table 2.4.: Representation of the elements in a software system using the city metaphor

group represent a package. Therefore, the country does not visualize the packages anymore, but the bundles. Hence, the city metaphor is also a possible approach to display OSGi-based systems. [5] takes a closer look both the city and the island metaphor and compares them. Misiak describes the city metaphor as *not always intuitive*. This is

one of the reasons why IslandViz makes use of the island metaphor [5].
The other main aspect discussed in this thesis is the usage of a conversational interface. The integration of a conversational interface for controlling applications in virtual reality is the subject of recent researches [18][19]. Int [19], a three-dimensional VR system for managing and controlling data in air traffic is extended by voice recognition.
The addition of speech assistants to software visualization systems, however, is a new area of research. In [20], Bieliauskas and Schreiber examined the usage of a conversational interface for software visualization in an application in a two-dimensional environment. Adding a conversational interface to a software visualization system in a virtual environment, respectively in the present case in augmented reality, is a not widely explored area. This thesis aims to create new insights in this field.

# 3. Creating Hypotheses for the Evaluation of Usability

This thesis discusses the usability of the conversational interface of IslandViz. Different aspects of usability are therefore introduced and examined in this chapter. Besides the usability, the utility is briefly discussed as well. Both terms can be summarized in the term usefulness [21].
The ISO norm 9241-11 gives a definition of the term usability. Section 3.2 gives a detailed view on this norm.
Thereafter, three hypotheses are put forward. The evaluation of IslandViz will be carried out taking these hypotheses into account. To verify them, a user study is conducted. The user study is further discussed in the next chapter.

## 3.1. Utility and Usability

In [21, p. 24], Nielsen uses the term *usefulness* to summarize *utility* and *usability*. He defines usefulness as *the issue of whether the system can be used to achieve some desired goal*.
The term utility addresses whether the system provides the functionality that is needed for the system. As this thesis focuses on the usability of IslandViz, the utility does not represent the central aspect of the examination. However, the utility should still be considered when evaluating the results as it might give additional suggestions on how

to improve IslandViz.

For usability, there is no uniform definition, currently. While he describes utility to address whether a function is provided, Nielsen describes usability as how well users can use the functionality provided by a system [21]. The ISO standards 9126 and 9241-11 give definitions of usability as well. However, all definitions differ from each other. The different definitions considered in this work are shown in Table 3.1. In [22], Abran et

| Source | Definition of usability |
|---|---|
| Nielsen in [21, p. 25] | Usability applies to all aspects of a system with which a human might interact, including installation and maintenance procedures. |
| ISO 9126 | The capability of the software product to be understood, learned, used and attractive to the user, when used under specified conditions. |
| ISO 9241-11 | The extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use. |

Table 3.1.: Different definitions of usability

al. attribute the reason for the different definitions to the fact that each definition has a different perspective on the topic of usability. Abran et al. carve out three different audiences for a software system: end-users, managers and software developers. Each of these groups has a different viewpoint on usability.

- To the end-user, software usability is important as it increases his performance when working with the software system.
- Managers expect usability for a software system, as it enhances the learnability of the product and, therefore, leads to more customers consuming this product.
- Software developers interpret usability as attributes like design quality and documentation maintainability.

As the each group has a different view on usability, their definitions differ from each other. In this work, the definition of the viewpoint of the managers has to be inspected closer. As DLR, which is responsible for the development of IslandViz, is a research institute, it does not try to sell its products to consumers. Nevertheless, the projects managers have a similiar viewpoint on usability. They want to justify their research in this area and, therefore, learnability is crucial to give the executives a quick impression of why this system is being developed.

The definitions listed in Table 3.1 cover different aspects of usability. Nielsen's definition leaves a lot of room for interpretation as he does not address concrete aspects. The ISO norms focus on concrete aspects, such as learnability and effectiveness. While the ISO 9126 was defined by a group of software engineering experts, the ISO 9241 was defined by a group of experts in Human Computer Interaction [22]. Thus, the definition of usability from the ISO 9241 will be used over the course of this thesis. The evaluation carried out in this thesis focuses on the interaction between IslandViz, respectively the conversational interface, and the user. Furthermore, this definition gives concrete aspects to examine and evaluate. Therefore, this definition of usability is fitting for the problem and is further discussed in this work.

There are many more definitions of usability, e.g. by IEEE [22] and a definition by Nielsen [21, p. 26] which is more detailed than the one given in Table 3.1. Each definition uses different approaches to explain usability and have their advantages and disadvantages. The ISO 9241-11 is a compact definition developed by people specialized in the field of Human Computer Interaction. Therefore, this portrays a fitting foundation for evaluating IslandViz.

## 3.2. Analyzing the Definition of Usability

As shown in Table 3.1, the ISO 9241-11 is defined as follows:

*"The extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use."*

This definition contains several variables. In this case, the *product* refers to IslandViz. The term *specified context* refers to navigating through IslandViz. Thus, *specified goals* refers to finding a particular component or obtaining information about it. The terms *effectiveness*, *efficiency*, and *satisfaction* indicate the aspects for which IslandViz needs to be examined.
In [23], effectiveness, efficiency and satisfaction are defined as follows:

- Effectiveness: the accuracy and completeness with which users achieve specified goals.
- Efficiency: the resources expended in relation to the accuracy and completeness with which users achieve goals.
- Satisfaction: freedom from discomfort, and positive attitude to the use of the product.

When evaluating IslandViz, all three aspects are considered. The application will be examined in regards to three hypotheses, which will be introduced in the next section.

## 3.3. Putting Forward the Hypotheses

IslandViz is examined to evaluate whether the addition of a conversational interface is successful in supporting the user navigating through the application. Therefore, three hypotheses are put forward. The goal of the evaluation is to either verify or negate the hypotheses. Based on these results, a conclusion can be drawn as to whether adding the speech assistant is successful in improving the usability of IslandViz. The evaluation considers the aspects of effectiveness, efficiency, and satisfaction. Each hypothesis focuses on a different aspect. Together, the hypotheses are to be used to answer the initial research question, whether a conversational interface improves the usability in IslandViz.

### Effectiveness

**H1** Adding a conversational interface to IslandViz does not affect the user's effectiveness in performing tasks.

In this case, the term effectiveness refers to the amount of activities a user needs to perform a task. The fewer activities a user has to perform for a certain task, the higher the effectiveness. To calculate the effectiveness, however, the term activity needs to be defined.

The definition of an activity is dependent on whether the user uses gesture control or voice control. Using gestures, an activity is a single gesture. As described in Section 2.4, IslandViz supports three different gestures, the Air-Tap, the Tap-and-Hold and the Two-handed Tap-and-Hold. In the context of voice control a spoken sentence represents an activity, as long as the NLU Service is able to recognize the intent.
Similar to other researches [24][25], only activities that are recognized by the system are taken into account for evaluating the effectiveness.

## Efficiency

**H2** The completion of tasks in IslandViz via a conversational interface takes less time than with the use of gesture control.

While the first hypothesis focuses on the amount of activities necessary to perform a task, this hypothesis takes a look on the time spent to perform the same task. The time considered is the time that has passed from the beginning of the task to its end. Hence, activities not recognized by the system are implicitly taken into account as well, as they occur during this time span. H1 however only refers to the activities recognized by the system. The difference in both approaches must be considered when evaluating the hypotheses.

## Satisfaction

**H3** The addition of a dialog interface affects user satisfaction.

Contrary to the effectiveness and the efficiency, satisfaction can not be measured as a number by the results of the tasks. Thus, another metric is required. A common approach to measure the satisfaction is a questionnaire. Section 4.6 deals with measuring the satisfaction and whether a questionnaire is used.

To evaluate the hypotheses, IslandViz must be tested. The goal of this testing process is to receive information which can be displayed on a metric. The information can then be used to support or refute the hypotheses and ultimately answer the research question. Several tasks are created to test the effectiveness and the efficiency. For measuring the satisfaction, a questionnaire is set up.

# 4. Study Concept

The hypotheses serve to examine the different aspects of usability defined in ISO 9241-11. The effectiveness and the efficiency can be measured, as the hypotheses imply the metrics time and amount of actions. To evaluate the usability of IslandViz, a group of users is selected. They are asked to perform several tasks using either gesture control or the conversational interface. The results of this examination is then taken into consideration during the following evaluation.

The examination serves to evaluate whether the conversational interface improves the usability of the navigation in IslandViz. Therefore, it compares the navigation with gesture control to the navigation with an additional speech assistant. To make a comparative study possible, the different elements of gesture control and voice control are compared with each other. The tasks within the user study are chosen based on the result of this comparison.

Contrary to effectiveness and efficiency, satisfaction is not measured by asking users to perform several tasks. Instead, the users use the system and are then asked about their view on the system. This can be achieved by handing out a questionnaire [21]. Another approach to measure the satisfaction is by giving the users the choice whether they use gesture control or the conversational interface to execute a certain task, after they have already worked with the system for some time. Both options are weighted at the end of this chapter.

Section 4.1 discusses the profile of the people that are asked to participate in the user study. Next, the comparison of the different control types is described. New intents are added in Section 4.3 to the NLU Service to make a comparative study possible.

Section 4.4 compares two different ways to design the user study. Section 4.5 focuses on the creation of the different tasks carried out during the study. Finally, Section 4.6 discusses the options for measuring user satisfaction and which method is included in the user study.

## 4.1. Target Group

IslandViz is an application that focuses on a very specific topic, the visualization of software systems. Therefore, only relatively few people come into contact with this application. Thus, the group of people considered for the user study ought to meet several expectations.

IslandViz visualizes software systems by portraying its architecture via islands, buildings and, harbours. Thus, the target user is already familiar with both computer science and the concept of software architecture. Software systems which can be visualized by IslandViz must be based on OSGi. However, this thesis discusses the addition of a conversational interface to any sort of software visualization application, regardless of whether it is based on OSGi or not. IslandViz is used exemplarily, but the statement of this work aims to apply to any sort of visualization software. Therefore, the target user does not have to be familiar with OSGi. In the user study, however, the user receives a small introduction to OSGi to understand the representation of the different components in IslandViz.

IslandViz is designed to support the onboarding of new employees. Nevertheless, different use cases are possible in the future. For example, the visualization of software could be used during a review process. In this case, the user of IslandViz is likely to have experience with the visualized software. Hence, both users that are experienced and inexperienced in the visualized software are likely to use IslandViz. Hence, it is irrelevant, whether the user has worked with the visualized software already. Anyway, the target user is in any case supposed to be involved in developing the visualized software. These characteristics are represented in Table 4.1. The people participating

| **The person...** | **Yes** | **No** | **Irrelevant** |
|---|---|---|---|
| is familiar with OSGi | | | x |
| has knowledge in the field of Computer Science | x | | |
| is familiar with Software-Engineering / the concept of Software-Architecture | x | | |
| has already worked with the visualized software | | | x |
| is supposed to support the development of the visualized software | x | | |

Table 4.1.: Characteristics of the target user of IslandViz

in the user study must at least meet the 2nd and 3rd requirement. The last characteristic, that users of IslandViz are supposed to develop the visualized software, is not considered in the study. Otherwise, the number of possible participants would be too small and too few results would be obtained. Furthermore, it has no impact on the user study whether a user is actually supposed to support the software visualized. Hence, regarding the user study, it does not matter whether the user is involved in developing the software after using the application.

The number of people participating in the user study must also be specified. There are different opinions on how many people should participate. In [26], Nielsen states that the best results come from testing five users only. According to Nielsen, five users discover roughly 80% of the usability problems. If the test is performed with additional users, more design errors will be detected. However, the effort increases disproportionately. In [27], Hwang et al. go into this statement. According to their researches, five people are not sufficient to reach 80% discovery rate. They claim that the number of people participating in the study to detect the majority of usability issues must be about 10. Therefore they set up the $10 \pm 2$ rule. This means, that depending on the used method for the usability evaluation, eight to twelve users are required to discover

80% of the usability problems. This rule, however, refers to the so-called usability evaluation methods Think Aloud, Heuristic Evaluation, and Cognitive Walkthrough. The thesis on hand creates user tests to evaluate the usability of IslandViz. Hence, none of these three methods matches to the problem described in this thesis. Alroobaea et al. did another research and came to yet another conclusion. They set up the $16 \pm 4$ rule to specify the optimal number of participants to achieve the 80% discovery rate for user testing [28]. This rule also covers the statement from each of the previous presented studies that more users naturally discover more usability errors. Hence, this thesis makes use of the $16 \pm 4$ rule.

## 4.2. Gesture and Speech Control

Voice control does not offer new capabilities to IslandViz. Each action that can be performed using the conversational interface can be performed by using gesture control as well. Table 4.2 compares the activities of voice control with those of gesture control.

The first four actions reflect the aspect of exploration. Currently, gesture control is even superior to voice control in this aspect, as voice control does not offer the possibility to navigate through IslandViz. The presented concept of voice control allows the users to say the name of a specific bundle and the application navigates there. However, they can not move the camera around freely. This presents a problem, as the user study is aimed to be a comparative study, where each action can be executed by gesture and voice control. This issue is further discussed at a later stage in this section.
When selecting a unit via gesture control, the user needs to perform an Air-Tap. In this case, they need to focus the corresponding unit with the HoloLens before tapping. Only the focused unit is selected when performing the Air-Tap. In the case of voice control, the user needs to give an utterance like “select [name of the unit]” to select the corresponding unit. If they do not know the name, they can focus the unit to display its name. Thus, voice control has an advantage in selecting a unit, as the user can

| **Action** | **Gesture control** | **Voice control** (corresponding intent) |
|---|---|---|
| Zoom | Two-handed Tap-and-Hold | zoom_in, zoom_out |
| Navigate | Tap-and-Hold | - |
| Select a unit | Air-Tap | select_component |
| Deselect a unit | Air-Tap | - |
| Select the biggest or smallest unit | Examine all units that should be considered and select the largest or smallest of them | select_biggest_component, select_smallest_component |
| Receive information about a unit | Select the corresponding unit and check the panel | summarize_information |
| Count the number of units | Count the units manually | count_components |

Table 4.2.: Different executions for an action using gesture or voice control

search for a specific name. Using gesture control, the users have to navigate through the application until they find the unit they are looking for, as described in Section 2.2. This advantage presents a problem as well. The two elements compared in the user study must provide the same services. In the present case, both gesture and voice control offer advantages over each other that must be considered when setting up the tasks.

The last three actions described in Table 4.2 contain a logic. Gesture control does not provide an activity to perform these actions. Instead, the user has to do them manually, e.g. by counting units or comparing the size of different units. Voice control offers activities that take on the task. The user can give an utterance like "what is the number of classes in [name of the unit]" to perform the task of counting components. If they perform such tasks with speech control, the user will most likely achieve much better results than if they perform the task with gesture control. Using these actions in the user study would falsify the results. The intents were explicitly added to voice control to perform these actions, since gesture control does not provide such features. Hence, these actions do not represent features that can be compared within the user study. Accordingly, only the actions that focus exploring IslandViz are considered.

The actions used to explore IslandViz are the first four entries in Table 4.2 (Zoom, Navigate, Select a unit, Deselect a unit).

## 4.3. Adding New Intents

In its current form, the NLU Service can process seven intents. Only three of them are used for navigating through IslandViz application: zoom_in, zoom_out, and select_component. In terms of navigation, gesture control offers an advantage, as the user can use the Tap-and-Hold gesture to shift the ocean. Furthermore, the user can deselect a unit and go back to an upper level. Currently, neither of these actions can be performed using voice control.

Navigating through IslandViz is a crucial part of exploring a software system. As voice

control aims to be an improvement compared to gesture control, every function available using gesture control needs to be transferred to voice control. Hence, new intents are necessary. It is important to mention that these intents are not included to the NLU Service in order to carry out the user study. In fact, creating the user study showed, that the NLU Service is missing some fundamental intents and must therefore be extended.

Five intents are added to cover the functionalities. To shift the ocean into different directions, the intents move_left, move_right, move_up, and move_down are included. From here on these are called *movement intents*. One could assume that this approach would have the disadvantage, that the user can only navigate in four different directions, while gesture control allows the user to move into any possible direction and at any angle. However, Section 5.1 shows, that the proposed implementation is capable of moving the visualization into any direction as well. Hence, both gesture control and voice control offer the same capabilities in terms of navigating through IslandViz.

The fifth intent added is deselect_component. As already mentioned, this intent is used to deselect the currently selected component and switch to a prior state.

With these changes, the NLU Service provides every intent necessary to navigate through IslandViz. Table 4.3 shows the new intents inserted in Table 4.2. The updated table solely includes the actions considered in the user study, that is, the actions related to navigation.

## 4.4. Study Design

The study described in this thesis compares the navigation via gesture control to speech control. To do this, three tasks are set up that each user is ought to perform. Each task is performed both with speech and gesture control. For the execution of the user study, two different designs are possible. In a study based on the *within-subject design* users perform each task using both navigation types. Hence, they perform each task twice. The *between-subject design* states that a user is assigned to one of two groups.

| **Action** | **Gesture control** | **Voice control** (corresponding intent) |
|---|---|---|
| Zoom | Two-handed Tap-and-Hold | zoom_in, zoom_out |
| Navigate | Tap-and-Hold | move_right, move_left, move_up, move_down |
| Select a unit | Air-Tap | select_component |
| Deselect a unit | Air-Tap | deselect_component |

Table 4.3.: Different executions for navigation actions using gesture or voice control

One group performs the three tasks using gesture control, the other using voice control. A disadvantage of the within-subject design is the learning process. After executing a task using gesture control, the user is aware of the situation and the solution of this task. So if users execute a task with voice control after performing the same task with gestures they are most likely faster than when executing the task with voice control for the first time. The learning process must not occur, as it falsifies the results. In the case of IslandViz, this phenomenon can be circumvented by choosing different islands for the same tasks based on the selected control type. For example, in the first task, the user must navigate to a preselected island. Based on whether the task is performed via speech or gestures, the users must select different islands. Consequently, the learning process is circumvented.

One might argue, that in different tasks, the user does not fully understand the task itself before starting it. In this case, the learning effect occurs again. Users finish a task and then know the way how to solve it. If they are then prompted to execute the task again, they show a better performance than in their first try. To prevent this, the tasks are designed to be short and easy to understand. Furthermore, the order in which way a task is executed is randomized. Some users perform a task using gesture control first and voice control afterwards. In the other cases, the order is the other way around.

With the learning effect being prevented, the evaluation can be designed based on the within-subject design. The main advantage of this design over the between-subject design is the creation of larger amounts of data. In the case of a between-subject design, two groups are created, where each group consists of $\frac{1}{2} \cdot n$ users, where $n$ is the total number of users. Hence, only $\frac{1}{2} \cdot n$ results can be established for each control type. Using the within-subject design, $n$ results are established for each control type.

## 4.5. Setting Up the Tasks

Three different tasks are set up for the user study. As the study makes use of the within-subject design, each task is implemented with two different solutions, one solution for gesture control and one for voice control. This section focuses on the different tasks and which solution is chosen for which control type.

### Demo

Before executing the tasks, the users can experience IslandViz using a Demo environment. Here, the users can get to know all actions, both for gesture and speech control. Before executing the tasks, the users are introduced to this environment so that they get comfortable using the different control types and actions. When starting to perform the tasks, the user should focus solely on performing the tasks, not on learning the control types.

### First task: Select the highlighted island

In the first task, the view on the application is zoomed out so that every island of the visualized software is visible. A particular island is highlighted. In order to perform this task the user must navigate to this island and select it.
This task is easy to perform and serves as an introduction for the user to understand how the different tasks are carried out. Additionally, navigating through the application and finding a particular island is a main part of IslandViz.
When performing the scenario using gesture control, the island that is highlighted and must be selected is called *RCE Components Switch GUI*. Using voice control, *RCE Database Component Execution* is the island to select.
Table 4.4 highlights the actions the user is expected to perform. Zoom and Navigate

| **Action** | **Gesture control** | **Voice control** (corresponding intent) |
|---|---|---|
| Zoom | Two-handed Tap-and-Hold | zoom_in, zoom_out |
| Navigate | Tap-and-Hold | move_up, move_down, move_right, move_left |
| Select a unit | Air-Tap | select_component |
| Deselect a unit | Air-Tap | deselect_component |

Table 4.4.: Estimation of the actions used to perform the first scenario

is necessary to navigate to the correct island. After navigating, the island must be selected.

### Second task: Find the smallest building on a particular island

On the start of this task, the view on the application is the same as in the preceding task. Again, an island is highlighted. The island contains five buildings of different

sizes. This time, users must not only navigate to and select the island. To finish the task, they must name the smallest building. As three out of the five buildings contain about the same number of lines of code (LoC), their height is the same. Thus, users must select each of the three buildings and check their LoC on the information panel. Only then, they are able to name the smallest building.
The islands chosen for this task are called *RCE XML Loader Component GUI* for gesture control and *RCE Excel Component Execution* for voice control. Table 4.5 shows the number of LoC for each class within the two islands.

| RCE XML Loader Component GUI | | |
|---|---|---|
| Name building | # LoC | # floors |
| XMLLoaderInputs-OutputsSection | 38 | 2 |
| XmlLoaderCompo-nentSection | 146 | 4 |
| XmlLoaderCompo-nentFilter | 34 | 2 |
| XmlLoaderHistory-DataItemSubtree-Builder | 97 | 3 |
| Messages | 41 | 2 |

| RCE Excel Component Execution | | |
|---|---|---|
| Name building | # LoC | # floors |
| ExcelComponent-Validator | 182 | 4 |
| Messages | 64 | 3 |
| ExcelRCECompo-nent | 101 | 3 |
| ExcelComponent | 363 | 6 |
| ExcelPersistent-ComponentDescrip-tionUpdater | 88 | 3 |

Table 4.5.: Number of lines of code per building on both islands

In RCE XML Loader Component GUI, the three buildings with the same height have two floors each and the smallest building, *XmlLoaderComponentFilter*, contains 34 LoC. In *RCE Excel Component Execution*, three buildings are made up of three floors. The smallest of these buildings, *Messages*, consists 64 LoC.

In this task, the user is expected to use the same actions as in the preceding task. However, the user must use the action Select more often, as several buildings are to select and compare. The Deselect action is not necessary to complete this task. When investigating a building, the user can use the Select action to just select another building, without explicitly deselecting the current one.

### Third task: Select the island that is furthest away from the starting island

In this task, the users start with a different view than in the other two tasks. An island is preselected and they have a close-up view on that island. The task is to find and select that island, which is the farthest away from the island which is selected in the beginning. To do so, the user must first deselect the preselected island and zoom the view out. The searched island is not highlighted. However, the islands are arranged in a way, that it is obvious for the user, which island is the farthest away. The preselected island is highlighted the whole until the task is finished.
This task requires the users to use every intent available.

## 4.6. Examining the Satisfaction

In order to measure satisfaction, the user is presented with several questions to answer or several statements to evaluate. The first three questions ask about the overall impression by the user about the system:

- *Please indicate which type of navigation (voice control or gesture control) you prefer.*
- *How easy did you find it to solve the tasks with gesture control?*
- *How easy did you find it to solve the tasks with voice control?*

The first question is answered by asking on of the three options *Speech control*, *Gesture control*, or *undecided*. The following two questions are answered by rating them with a value from 1 (very easy) to 5 (very difficult).
Furthermore, a standardized questionnaire, the SUS (System Usability Scale), is used. This scale gives ten statements the user has to rate on a scale from 1 (Strongly Disagree) to 5 (Strongly Agree). The items 1, 3, 5, 7, and 9 give positive statements, while the other items are formulated negatively[29].

1. *I think that I would like to use the system.*
2. *I found the system unnecessarily complex.*
3. *I thought the system was easy to use.*
4. *I think that I would need the support of a technical person to be able to use the system.*
5. *I found the various functions in this system were well integrated.*
6. *I thought there was too much inconsistency in the system.*
7. *I would imagine that most people would learn to use this system very quickly.*
8. *I found the system very cumbersome to use.*
9. *I felt very confident using the system.*
10. *I needed to learn a lot of things before I could get going with this system.*

In the present context, the term *the system* is not clear, as the evaluation compares two different control systems. Hence, each user answers the ten questions twice. In one case, the term *the system* is replaced by *the gesture control in IslandViz*, in the other case it is replaced by *the voice control in IslandViz*.
Each item of the SUS is weighted by a value from 0 to 4. To do this, the value of each item formulated positively (1, 3, 5, 7, 9) is reduced by 1, whereas the other values are subtracted from 5 each. The resulting ten values are added together and multiplied by 2.5. This results in a value from 0 to 100. High values on the SUS scale indicate a good usability of the system.

# 5. Establishing Technical Conditions

To carry out the user study, IslandViz must be able to perform actions according to the intent recognized by the NLU Service. In its current form, IslandViz is already able to communicate with the NLU Service. However, the application can not process the information received by the service. Thus, new actions must be implemented, IslandViz can perform when it recognizes a certain intent. Furthermore, these actions must be mapped to the belonging intent. With these changes, the pipeline when processing an utterance looks as follows:

1. User says an utterance.
2. The system transforms the utterance into a string.
3. The string is sent to the NLU Service. This service extracts intent and entities and sends these information back to IslandViz.
4. IslandViz maps each intent to an action. Thus, the action belonging to the extracted intent is performed.

This chapter focuses on the fourth step, as the preceding steps are already implemented. In Section 4.3, five new intents are added to the NLU Service. These intents are necessary for navigating through IslandViz. As shown in Table 4.2 (Section 4.2), several actions that can be performed via gesture control can not be performed by speech control. Thus, the corresponding intents are added, so that these actions can be executed via voice control as well. Section 5.1 then focuses on the implementation

of these actions for speech control. In the cases of selecting and deselecting an island, existing functions can be reused. The method to select an island used by gesture control requires the user to focus the island that is to be selected. Hence, using this function for voice control means, that users cannot say the name of the island they want so select. Instead, they need to focus it and say an utterance like *Select this island*. Selecting an island by its name is not possible in the presented work. This, however, is not intended anyways, as it would give an advantage to voice control over gesture control.
The actions for zooming and moving the visualization must be implemented separately for gesture and speech control. Section 5.2 focuses on the mapping of intents to actions. Finally, Section 5.3 discusses the implementation of the tasks, set up in Section 4.5, in IslandViz.

## 5.1. Implementing Navigation Actions

Having every necessary intent specified, the according actions must be implemented in IslandViz. When performing a Navigate action with gesture control, the repositioning of the components is dependent on how far the user moves his arm when performing the Tap-and-Hold. With voice control, an arbitrary adjustment of the position of the components is not possible. Even though the Navigate action serves the same purpose for both gesture and voice control, it differs greatly in terms of the implementation. Hence, an additional implementation for the Navigate action has to be developed. According to this argumentation, the same goes for zooming in and out.
Implementing the actions for speech control can be divided into two different parts. The movement intents consist of four intents: move_up, move_down, move_left, and move_right. The implementation for these intents is discussed in Section 5.1.1. Section 5.1.2 covers the intents zoom_in and zoom_out. As mentioned above, the actions for the intents select_component and deselect_component can reuse the already existing functions. Hence, the actions do not have to be implemented again.

## Content Pane and Visualization

Within the next two sections, the terms *pane* and *visualization* are used frequently. The visualization describes every component visualized by IslandViz - islands, regions, the ocean, etc. are all summed up by this term. The Content Pane, which is from now on referred as by pane, is the panel which describes the position of each component of the visualization. The ocean lies within this panel, while the islands are placed on top of the ocean, hence, placed on the panel. The regions are placed on top of the islands and the compilation units, in turn, are placed on these regions. Ultimately, this makes the position of every component of the visualization dependent on the position of the pane.
The pane is also initialized with certain boundaries. Due to these boundaries, the pane has a certain height and width and, thus, spans an area. Only the components of the visualization which are placed within this area are shown to the user. This means, that by performing gestures like the Tap-and-Hold the user does not interfere with the pane, but with the visualization. By repositioning the visualization, a different area of it lies within the boundaries of the pane and is therefore displayed.

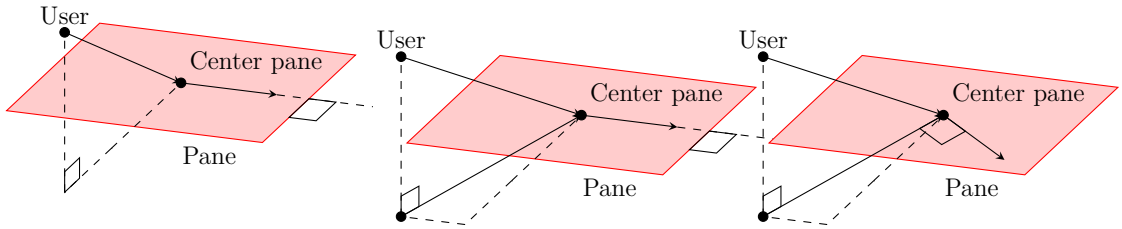
### 5.1.1. Shifting the View on the Application

Similar to dragging the visualization by performing a Tap-and-Hold, users should be able to say utterances like *move down*, *go right*, etc. to move it. However, a sentence like *go right* can be interpreted in two ways. By saying this utterance, users could expect the point-of-view on the application to go to the right. Hence, the part of the ocean to the right of the currently visualized part is shown. Another might expect the visualization itself to go to the right. In this case, the islands to the left of currently visualized part are shown. The following implementation for shifting the view is based on the first interpretation and therefore relative to the user's point-of-view.
The movement intents can also be interpreted differently, in a sense that the terms up,

down, left, and right are all relative. Hence, moving to the right can have two different interpretations.

- Move to the right relative to the user.
- Move to the right relative to the pane.

If the user stands exactly in front of the visualization, *right* means the same in both cases. However, if the user moves slightly in any direction, the term *right* from the user's point of view is different to the pane's. This is depicted in Image 5.1.




(a) Both interpretations lead to the same outcome.

(b) Move the visualization relative to the pane.

(c) Move the visualization relative to the user.

Figure 5.1.: Different interpretations of the action *Move to the right*

The figure to the left shows a user standing exactly in front of the pane. In this case, moving to the right is clear. In the two figures to the right, the user moved leftwards. If interpreted relative to the pane, *go right* makes the visualization move perpendicular to the pane (see Fig. 5.1b). Fig. 5.1c shows how the visualization is moved when the action is interpreted as relative to the user. In this case, the movement vector of the visualization is perpendicular to the vector between the user and the center of the pane.

For the user, it is more convenient, if the visualization moves relative to him. This also makes the navigation through the application more free. While the user only has four intents to move the visualization, he can walk around the application by a certain angle and then perform a Navigate action via speech control to change the angle in

which the visualization moves (compare 5.1a and 5.1c).

To calculate the vector by which the plane must be shifted, three different vectors must be taken into account.

```
1 // Get position information
2 Vector3 headPosition = GazeManager.Instance.GazeOrigin;
3 _contentPane =
     UIManager.Instance.GetUIElement(UIElement.ContentPane);
4 Vector3 contentPosition = _contentPane.transform.position;
5 Vector3 contentNormalVector = _contentPane.transform.up;
```

Listing 5.1: Store the different vectors

`headPosition` gives the position of the HoloLens, hence of the user's head. The other two vectors, `contentPosition` and `contentNormalVector` give the position and the normal vector of the panel which holds the visualization. These vectors are displayed in Fig. 5.2. This figure shows an abstract representation of the most

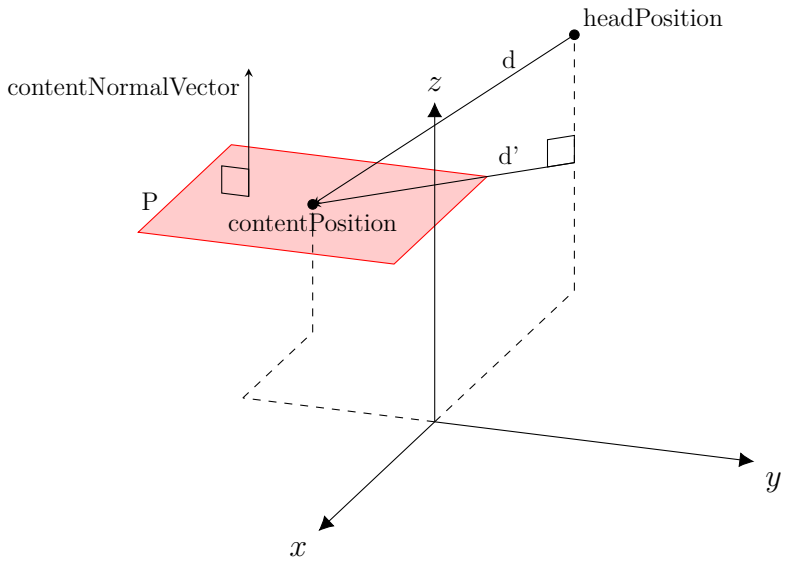



Figure 5.2.: Vector from the user to the visualization projected on the plane.

important aspects for the following calculation.

To calculate the vector in which the visualization is shifted after a moving intent is recognized, the vector between *headPosition* and *contentPosition*, depicted as $d$, is calculated. This vector is then projected on the panel and is called $d'$. For this projection, the code in Listing 5.2 is executed. Note, that in this code, vector $d$ is called `paneDirection` and $d'$ is denoted as `projectedDirection`.

```
1 // Project the vector from head to content into the plane.
2 Vector3 paneDirection = contentPosition - headPosition;
3 float heightFactor = Vector3.Dot(contentNormalVector,
     paneDirection);
4 Vector3 height = contentNormalVector * heightFactor;
5 Vector3 projectedDirection = Vector3.Normalize(paneDirection -
     height);
```

Listing 5.2: Project a vector into the pane

The method `Vector3.Dot` receives `contentNormalVector` and the vector between `headPosition` and `contentPosition` and calculates the scalar product. By multiplying the normal vector of the content pane, which is normalized per default, with the scalar product and subtracting this result from `paneDirection`, the vector `paneDirection` is sucessfully projected on the pane. The result is stored in the variable `projectedDirection`.

The projectedDirection gives already the direction for moving away from the user, which is addressed by the intent move_down. Multiplying this vector with -1 therefore, gives the direction for moving up. To calculate the vectors for moving to the left and right, the normal vector of the pane must be considered again. As projectedDirection lies within the pane of the normal vector, the cross product of them results in another vector within this plane. As these vectors are at a 90-degree angle to the projectedDirection, these are the desired vectors that go to the left and right. When calculating the cross product, it is important to keep the order of the vectors in mind,

as it is not commutative.

$\overrightarrow{projectedDirection} \times \overrightarrow{normalVectorPane} \Rightarrow$ Vector to the left of the user.

$\overrightarrow{normalVectorPane} \times \overrightarrow{projectedDirection} \Rightarrow$ Vector to the right of the user.

Using these vectors, the visualization can be shifted into the according direction. However, simply adding the calculated vector to the position of the visualization leads to an abrupt change in the users view. Within an instant, every point of the visualization would be repositioned. To make the shift of the visualization run more smoothly, a time frame is defined in which the shift is processed. Furthermore, a function is defined, which indicates the progress of the move process.
$f(x) = c \cdot x$ could be used as a function that puts the distance of the move process in relation to the time passed. Using this approach, the visualization is shifted with a fixed velocity $c$. In this solution, the movement of the visualization might seem unnatural, as its velocity goes up form 0 to $c$ in an instant and goes down to 0 again, after the shift is finished. A better attempt is to change the velocity during the move process. A common approach is to have a low velocity at the start and the end of the shift, while it rises in the time inbetween.

The function shown in 5.3 is called Sigmoid function. It is defined as $\sigma(x) = \frac{1}{1+e^{-x}}$. The value range of the Sigmoid function lies in the interval (0, 1). This can be interpreted as the percentage of the progress of the shift. The problem with this interval is, that the values are never 0 or 1. The function only converges to these values. However, for x = -6 and x = 6, the values are very close to 0 and 1. Therefore, this distance can be disregarded and the Sigmoid function is used to describe the move process, as it allows a smooth run from the start position of the visualization to the target position.
The course of the Sigmoid function is considered for the interval x = [-6; 6]. However, x describes the time and the time cannot start at x = -6. Furthermore, the interval has a length of 12, while the duration of the move process is dependant on a variable, denoted as $t_{duration}$. Therefore, the time passed since the beginning of the move pro-

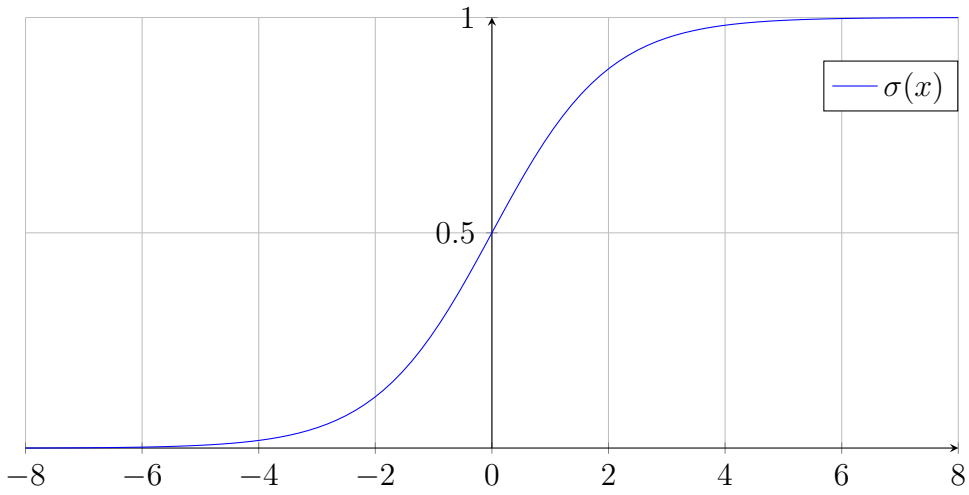



Figure 5.3.: Sigmoid function to progress the move process

cess, denoted as $t_{passed}$ must be adjusted before it is given as an input to the Sigmoid function. The goal of this adjustment is to map $t_{passed}$, which lies in an interval of [0; $t_{duration}$] to the interval of [-6; 6]. This is achieved by Equation (5.1):

$$x = f(t_{passed}) = (t_{passed} - 1/2 \cdot t_{duration}) \cdot 12 \cdot \frac{1}{t_{duration}} \tag{5.1}$$

x denotes the input for the Sigmoid function. At first, $1/2 \cdot t_{duration}$ is subtracted from $t_{passed}$. Therefore, the input interval is shifted, as seen in Equation (5.2).

$$[0; t_{duration}] \Rightarrow [-1/2 \cdot t_{duration}; 1/2 \cdot t_{duration}] \tag{5.2}$$

The input value is then multiplied by $12 \cdot \frac{1}{t_{duration}}$. This stretches the input interval to the desired input interval, as shown in Equation (5.3).

$$[-1/2 \cdot t_{duration} \cdot 12 \cdot \frac{1}{t_{duration}}; 1/2 \cdot t_{duration} \cdot 12 \cdot \frac{1}{t_{duration}}] = [-6; 6] \tag{5.3}$$

This means, that by passing the variable $t_{passed}$, which lies in the interval of [0; $t_{duration}$], into the function $f(t_{passed})$, the output lies in the interval of [-6; 6] and can be passed into the Sigmoid function.
Using Equation (5.1), the move process can be implemented in IslandViz. This is shown in Listing 5.3.

```
1  private IEnumerator ShiftPane(Vector3 startPosition, Vector3
       targetPosition) {
2      Vector3 currentPosition = startPosition;
3
4      float interpolation = 0.0f;
5      // Time passed since the beginning of the move process
6      float cumulativeTime = 0.0f;
7      float durationOffset = _duration / 2.0f;
8      float durationFactor = 12.0f / _duration;
9
10     while (Vector3.Distance(targetPosition, currentPosition) >
           0.001f) {
11         cumulativeTime += Time.deltaTime;
12         float sigmoidInput = (cumulativeTime - durationOffset) *
               durationFactor;
13         interpolation = Sigmoid(sigmoidInput);
14         currentPosition = Vector3.Lerp(startPosition,
               targetPosition, interpolation);
15         _visualization.transform.position = currentPosition;
16         yield return null;
17     }
18
19     _visualization.transform.position = targetPosition;
20 }
```

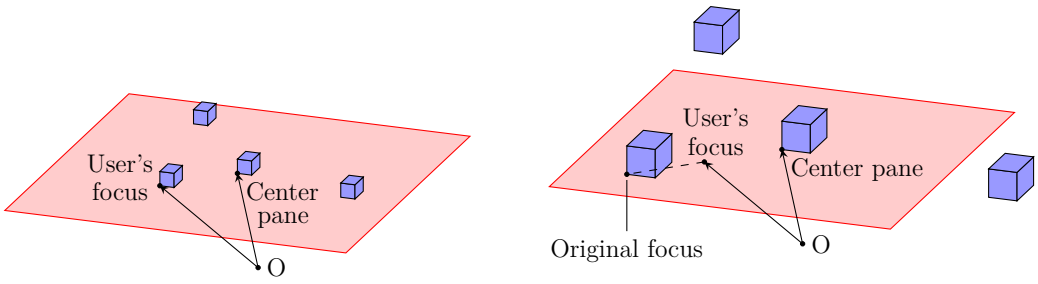
Listing 5.3: Shifting the visualization using Lerp

The variables defined in line 6 - 8 are used for mapping the time to the input of the Sigmoid function. The variable `interpolation` stores the output of the Sigmoid function. The reason for the name interpolation is explained in a later instant.
The while-loop is repeated, until the distance between targetPosition and currentPosition is lower than a fixed amount, here 0,001. Time.deltaTime denotes the time that

has passed since the last time, the while-loop executed this command. This value is added to cumulativeTime to calculate $t_{passed}$. Dependant on the time, the input for the Sigmoid function is calcuated in Line 13, while Line 14 calculates the according output. In Line 15, the command Lerp is used to calculate the new position of the visualization. It receives three parameters, startPosition, targetPosition, and interpolation. As it stores the result of the Sigmoid function, the interpolation has a value of (0, 1). Furthermore, it denotes the percentage, with what distance between startPosition and targetPosition was covered. If for example the Lurp function receives startPosition = (1, 0, 0), targetPosition = (2, 0, 3), and interpolation = 0.7, it returns (1.7, 0, 2.2) as a result. The result of the lurp function is stored in the variable currentPosition and is assigned to the position of the visualization in Line 17. The last method of this loop means, that this function, the ShiftPane function, is not further processed until the next frame is calculated.
After the while-loop is finished, the position of the visualization is set to the targetPosition. This step is processed, as the Sigmoid function only converges to 1, but never actually results in the value of 1.
After executing the ShiftPane method successfully, the visualization is shifted properly relative to the user's position.

### 5.1.2. Zooming into the Application

The terms zooming in and out refer to a change in the scale of the visualization. However, rescaling the visualization is always relative to the center of the visualization. This means, that if the user looks at an island and says *zoom in*, the application would actually zoom to the center, if only the scale is changed. This is depicted in Figure 5.4. The red plane represents the pane and the different cubes can be interpreted as an abstract representation of islands in IslandViz. The image displays the position of the center of the pane and the user's focus relative to the origin of the coordinate system. After changing its size, the visualization is updated as shown in Fig. 5.4b. Originally, the user focused a cube and performs the action to zoom in. As the application zooms




(a) View on the visualization

(b) The same view after zooming in without adjusting the position of the visualization's center

Figure 5.4.: Zooming without adjusting the center of the visualization

relative to the center of the pane, the cube is shifted to the left and therefore leaves the user's focus. Figure 5.4b shows both the position of the point that was originally focused and the point that is focused after zooming into the view. The proposed Zoom action is not intuitive, as the users expect IslandViz to zoom into the position where their focus lies. In order to to this, the visualization must not only be enlarged, but its position must be changed as well.

To achieve this, the coordinates of the focus must first be adjusted, so that it lies within the pane. If the user focuses the ocean, the focus is in the pane anyways. However, if an island is focused, it is outside of the pane. In this instance, the position vertically underneath is chosen. This is shown in Image 5.5. In this image, the vector to the

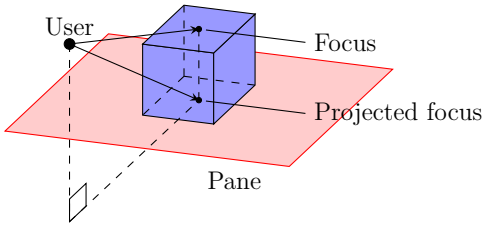



Figure 5.5.: Adjust the user's focus to a position on the panel

position of Focus is projected into the pane, just like in Section 5.1.1. The resulting position is named Projected focus.

Fig 5.6 shows a pane and a cube, which is located at the position of $\overrightarrow{Focus1}$. In the

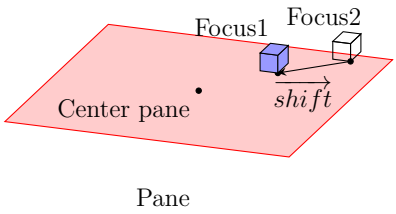



Figure 5.6.: Shifting the application back after zooming

present case, $\overrightarrow{Focus1}$ depicts the position of the user's focus. The contours of this cube are located at the position of $\overrightarrow{Focus2}$ and display the position to where the cube is shifted after zooming into the application. To compensate this shift, the visualization must be shifted by the vector $\overrightarrow{shift}$. Then, the cube is located at the focus of the user again. To calculate this vector, the calculations in Eq. 5.4 are performed.

$$\begin{aligned} \overrightarrow{d1} &= \overrightarrow{Focus1} - \overrightarrow{CenterPane} \\ \overrightarrow{d2} &= \overrightarrow{Focus2} - \overrightarrow{CenterPane} \end{aligned} \tag{5.4}$$

$$\overrightarrow{shift} = \overrightarrow{d1} - \overrightarrow{d2}$$

During the Zoom action, the visualization is moved by $\overrightarrow{shift}$. This compensates the shift of the focused position that happens naturally when changing the scale.
Similar to shifting the visualization, zooming makes also use of the Sigmoid function to perform the action over a certain time frame. The code described in Listing 5.3 can be reused, expect that the visualization must be enlarged. Therefore, the two lines in Listing 5.4 are added in the while-loop right before executing the yield-command.

```
1 currentScale = Vector3.Lerp(startScale, targetScale,
     interpolation);
2 _visualization.transform.localScale = currentScale;
```

Listing 5.4: Adjust the scale of the visualization

The startScale refers to the scale of the visualization before performing the Zoom action. The targetScale is dependent on whether a user wants to zoom in or out. After

the loop has finished, the scale of the visualization is set to the target scale, similar to its position being set to the target position.

After implementing the actions for the moving and zooming intents, every action necessary to carry out the user study is implemented, as the actions for selecting and deselecting already exist. Next, the intents extracted by the NLU Service must be mapped to the according action.

## 5.2. Map Intents to Actions

IslandViz already contains the enum KeywordType. This enum manages every action that can be performed by speech control. Besides from the actions, it contains two additional entries, `None` and `Invariant`. The purpose of these two intents is explained later in this section. Currently, the KeywordType only contains only the entries None and Invariant, as the intents have yet to be added to IslandViz. As described in the previous sections, IslandViz is supposed to perform eight actions. Therefore, the enum KeywordType is extended by the entries `Select`, `Deselect`, `ZoomIn`, `ZoomOut`, `MoveUp`, `MoveDown`, `MoveLeft`, and `MoveRight`.
Next, these KeywordTypes are linked to the according intents. This is done in the class NLUServiceClient. This class is responsible for communicating with the NLU Service. In Listing 5.5, a Dictionary is created, which maps every intent to a KeywordType.

```
private Dictionary<string, KeywordType> _intentToKeyword = new
    Dictionary<string, KeywordType>()
{
     { "zoom_in", KeywordType.ZoomIn},
     { "zoom_out", KeywordType.ZoomOut},
     { "move_up", KeywordType.MoveUp},
     { "move_down", KeywordType.MoveDown},
     { "move_left", KeywordType.MoveLeft},
     { "move_right", KeywordType.MoveRight},
     { "select_component", KeywordType.Select},
     { "deselect_component", KeywordType.Deselect}
};
```

Listing 5.5: Mapping the intents to a KeywordType

IslandViz is now capable of receiving an intent and transforming it into the according KeywordType. Next, these KeywordTypes must be mapped to an action. This is accomplished by using the Command class, which already exists in IslandViz.
As explained in Section 2.4, IslandViz uses a state manager. The meaning of a gesture is dependant on the current state. Hence, a command must be defined in an according class. The commands for moving are defined in the function `Init_StateMain` in the class AppManager. The AppManager is responsible for processing commands triggered by gesture or speech input. Such a command is defined as shown in Listing 5.6.

```
Command command_moveUp = new Command(GestureType.None,
    KeywordType.MoveUp, InteractableType.Invariant);
```

Listing 5.6: Definition of a Command object in IslandViz

The proposed command has the following properties:

- `GestureType.None` - As mentioned above, the KeywordType enum contained the entries None and Invariant be default. In fact, every enum which is processed by a command contains these entries. As this command is supposed to be executed when the user does not perform any gesture, the GestureType is set to None.

- `KeywordType.MoveUp` - The user used speech input and the NLU Service recognized an intent that is linked to the KeywordType MoveUp. As shown above, the belonging intent is move_up.

- `InteractableType.Invariant` - Invariant indicates, that the entry for this enum is not taken into consideration when processing this command. Hence, this enum can have any value. The enum InteractableType refers to the item focused by the user. Thus, the enum InteractableType contains entries like bundle, package, etc.

Finally, this command is linked to an action as shown in Listing 5.7.

```
1 state_main.AddInteractionTask(command_moveUp, task_moveDirection);
```

Listing 5.7: Linking commands to tasks

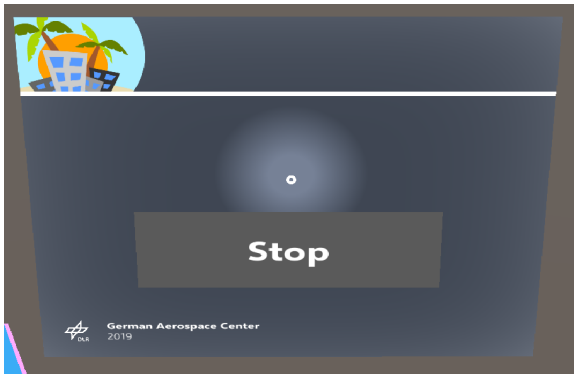
Due to this, the command command_moveUp is linked to the class task_moveDirection. Thus, when the command command_moveUp is performed, IslandViz executes the code within the class task_moveDirection. Both the command and the interactionTask must be added for the other three intents move_down, move_right, and move_left as well.
This command can only be used within the state Main, as the method `AddInteractionTask` is executed by the object `state_main`. If this command is supposed to work in other states as well, the `AddInteractionTask` must be executed with this command on each state that is supposed to be able to process this command.

By linking a command to a task as presented in Listing 5.7 the task `task_moveDirection` is executed whenever `command_moveUp` is recognized while IslandViz is in the state `state_main`.
With these changes, voice control is successfully implemented in IslandViz for the eight intents described in this chapter. These changes are going to stay in IslandViz after carrying out the user study, unlike the changes described in the next section. This section is about implementing the different scenarios, which are only necessary for the study.

## 5.3. Adding the Scenarios

After adjusting the position of the Content Pane, the user can usually start navigating in IslandViz. To carry out the study, IslandViz is to be extended so that between confirming the position of the pane and actually using IslandViz, a new panel appears. In this panel, the users can choose which scenario they want to perform and which gesture control they use. With this panel, the scenario panel, a new state needs to be added to IslandViz. Placing the Content Pane is performed in the state *Setup*. Afterwards, the users progresses to the state *Main*, where they can start to explore the visualized software system. The new state is put in between the two states and is called *Scenario*.
A scenario panel is to be shown while the user is in the Scenario state. Therefore, the methods OnStateEnter and OnStateExit show and hide the panel. Furthermore, the OnStateExit method extracts the settings the users set within the scenario panel. That is, which scenario they want to perform and whether they want to use gesture or voice control.
The new panel is shown in Fig. 5.7a. It consists of a dropdown menu, where the user can choose the next scenario, and two radio buttons, to choose between gesture and voice control. When pressing the Start button bottom right, the state of IslandViz changes from Scenario to Main and the scenario starts. Additionally, a stop button is


(a) Panel to choose the scenario from




(b) Stop button to finish the current scenario

visible while being in either the Main, Inspect island, Inspect region, or Inspect building state (see Fig. 5.7b). The user can press this button to go back to the Scenario state.

### 5.3.1. Implementing the Scenarios

As explained in Section 4.5, the user can choose between three tasks and a demo level. In the first scenario, the view is zoomed out, so that every island is visible. This is portrayed in Fig. 5.8. On the upper side of the pane, one island is highlighted and the user is asked to navigate to and select this island. Depending on whether the user performs this scenario via gesture or speech control, a different island is highlighted. Due to this, a learning effect is prevented. This is important, as the user is based on the within-subject design, which means, that the user performs the first task both with gesture and voice control. The same applies to the second scenario, where the user, again, has to navigate to a specific island and select the smallest building. In the third scenario, the island which is selected in the beginning is highlighted troughout the whole task. Similar to the first two scenarios, different islands are selected in the beginning, depending on the type of control.

A new class is added to IslandViz, the `ScenarioHandler`. This class serves as a central unit to manage the scenarios. After pressing the Start button on the scenario

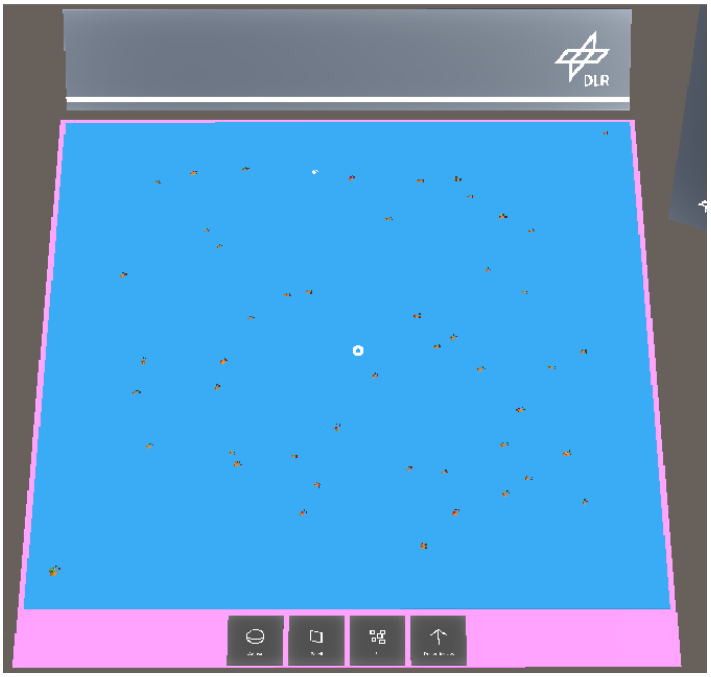



Figure 5.8.: Starting view for a scenario

panel, the ScenarioHandler receives the information extracted from the panel and processes them. Based on the chosen scenario and control type, the according island is highlighted. Furthermore, the ScenarioHandler tracks both the time and the amount of activities the users need to perform the tasks. At the end of each scenario, the ScenarioHandler sends these information to an additional service, which uses Flask to create a REST interface as well.

Instead of a user utterance like the NLU Service, this service receives a JSON which is structured as shown in 5.8.

```
1  {
2    "timestamp": "2019-8-26 - 13:35:42",
3    "data": {
4      "NumberScenario": "FIRST",
5      "TypeScenario": "VOICE",
6      "StartTime": "828.9665",
7      "EndTime": "909.8589",
8      "NumberActivitiesGesture": "0",
9      "ActivitiesGestures": [],
10     "NumberActivitiesVoice": "2",
11     "SpeechGestures": [
12       "Zoom[ZoomIn]",
13       "Select[Island]"
14     ]
15   }
16 }
```

Listing 5.8: Exemplary JSON as the result of a task

The file includes some redundancies, as it states that the user used voice gesture (see line 5) but still tracks the number of activities performed by gesture. This redundancy serves as a validation of the file. The values for StartTime and EndTime is the number of seconds that has passed since the start of IslandViz. The StartTime is set, when the user presses the start button on the ScenarioPanel, while the EndTime is set, when the stop button is pressed. The difference of these two values then gives the amount of time the user needed to perform this scenario.

As mentioned above, this JSON is sent to a Python service using the Flask interface. This service then stores the data in a file. For each run of a user, including all six tasks, a new file is created.

# 6. Evaluating the Results

After implementing the functions to navigate through IslandViz and including the different tasks, the user study can be carried out and the results evaluated. These results are taken into account to value IslandViz' usability. The tasks are designed to evaluate the effectiveness and the efficiency, while the questionnaire is used to rate the satisfaction. At the end of the questionnaire, users can leave a free text as feedback on the application. This feedback is considered for the future development of IslandViz as well.

## 6.1. Carrying out the User Study

16 users participated in the user study. Each participant went through the same procedure: A short introduction about the topic was given (see Appendix A[1]) as an opening text within the questionnaire. This text is about the use of the user study and how it is carried out. On the next page of the questionnaire, the users were asked to enter some demographic data, in the presented case age and gender. Next, four questions were put forward:

[1] As the user study was carried out with german participants, the texts within the questionnaire were written in german as well. The appendix contains both an original version of the text in german and a translation in english.

- *I have already worked with OSGi.*
- *I have already worked with RCE.*
- *I have used the HoloLens before.*
- *I have already used the implementation of IslandViz on HTC Vive.*

This information was requested to find out about possible interrelations between the foreknowledge about the presented software (RCE) and the performance of a user in the tasks. The fourth question refers to the implementation of IslandViz in VR (see 2.4). After answering these four questions, the first part of the questionnaire was finished.

Before performing any task, the users were introduced to the usage of the different controls in the Demo. In this scenario, the users could navigate through IslandViz in the usual way. This means, they were able to get familiar with both gesture and voice control before executing the scenarios. Furthermore, the application itself was introduced. Emphasis was put on the fact that the information about the lines of code is shown when the corresponding building is selected, as this information was necessary to perform the 2nd scenario. After receiving these information, the users were allowed to use IslandViz within the Demo mode as long as they please to grasp the concepts of IslandViz and the different controls. As soon as they felt comfortable with the navigation, the users could go on and start executing the different scenarios. The scenarios are described detailed in Sec. 4.5.

Next, the users were asked to perform the different tasks. Each task had to be performed twice. Once with gesture and once with voice control. The first scenario was always executed first. The order of the tasks second scenario and third scenario was randomized. Whether the user performs a task with gesture or voice control first is randomized for every scenario as well. The randomization served to decrease possible learning effects.

After finishing the tasks, the users were asked to finish the remaining part of the questionnaire. In this part of the questionnaire, three general questions and 20 specific

questions (the latter are based on the SUS scale) are asked. A more detailed description can be found in Sec. 4.6.
Having the questions answered, the questionnaire is filled out completely and the user study was finished. The whole process took approximately 30 - 45 minutes.

## 6.2. Evaluating the Different Aspects of the Usability Study

The user study is split into two main parts. The first part is the execution of the different scenarios. This part covers the aspects of efficiency and effectiveness, which are two of three aspects in regards of usability. The third aspect, satisfaction, is examined in the second part of the study, the questionnaire. The following subsections evaluate the results of both the first and the second part of the user study. Then, a brief summary is concluded about the different results and whether the aspects of efficiency, effectiveness, and satisfaction are related to another. Ultimately, the hypotheses, put forward in Sec. 3.3, are evaluated and the initial research question are discussed.

### 6.2.1. Examining the Results of the Tasks

During the user study the time and the amount of activities the users needed to complete each task was recorded. When performing the scenarios with gesture control, the users needed less time on average than when using voice control. Fig. 6.1 compares the amount of time the users needed to perform the different tasks.

The figure shows six different plots, where the plots are always grouped by two. Each pair of plots represents the results of one scenario. One plot shows the results for gesture control, the other for voice control.
The plot to the left in Fig. 6.1 is labeled as T1: Gesture. This means, that this plot displays the 16 measured time values to finish the first task (hence, T1) where the users

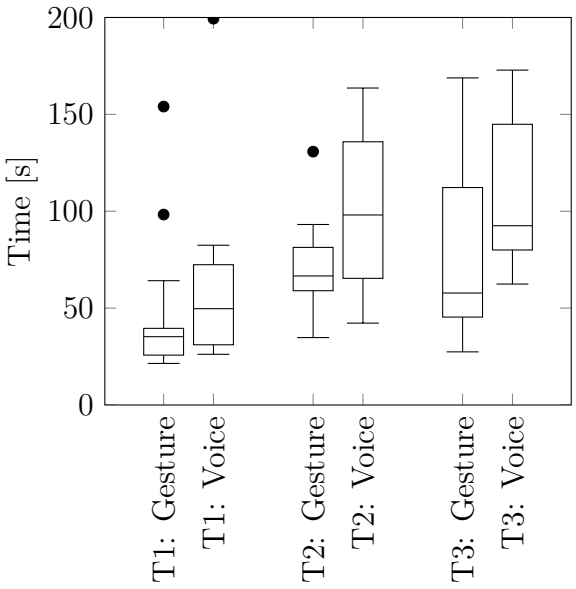



Figure 6.1.: The times the users spent performing the six tasks.

used gesture control. The two dots above the plot indicate outliers. These results were not taken into consideration during the evaluation. The outliers were caused by technical problems with voice control or the user heavily struggling to cope with gesture control. Therefore, these measured values were not representative for the user study.
On average, the users finished each task faster with gesture control than with voice control.
For both gesture and voice control, the users were faster finishing the first task. This was expected, as the second task includes the first task (navigating to a specific island). For voice control, both plots look similar. The second plot is larger than the first one. The relations of whiskers and interquartile range box, however, have remained roughly the same. This can be explained by the delay between the user demanding the utterance and IslandViz executing the action. As IslandViz uses different services to process a statement, a delay is inevitable. Furthermore, the speech recognition did not always work. This is time costly as well, as the user had to realize that his utterance was not recognized and then repeat the sentence. Since this task required the user to perform multiple actions, hence, say multiple sentences, these time frames added up. As the second task required more actions to be performed than the first one, more time frames

added up. Hence, the plot T2: Voice is higher on the vertical scale and larger in general than T1: Voice. These time frames do not occur when using gesture control, as the application reacts almost immediately to the user's gestures. Hence, gesture control shows a better performance in terms of efficiency.
While the first two plots look similar for voice control, the plots for gesture control are different. In the first task, 75% of the users needed less than 40 seconds, while the other 25% needed 40-70 seconds. A reason for this could be that the users who achieved the results slower than 40 seconds were still struggling with the control in the first task. This also explains the two outliers, who achieved results of 100 and 150 seconds. This interpretation is also supported by the fact, that 11 of the 16 users have already used the HoloLens at least once before. Gesture control was not completely unfamiliar to them. Hence, they achieved better results than the 5 who had never used the HoloLens before.
As the second task always takes place after the first one, the users are likely to have improved in navigating to a specific island. Hence, 75% of the users spent between 60 and 100 seconds to perform this task. The others might be especially experienced using HoloLens and, therefore, achieved better results than the majority.
The plot of T3: Gesture looks similar to T1: Gesture. The lower whisker is relatively small, the interquartile range box has roughly half of the size of the plot, and the upper whisker is relatively large. Both tasks were about navigating and mainly used the actions Navigate and Zoom. Some users struggled especially with the Two-handed Tap-and-Hold, hence, the Zoom action using gestures. This might explain, why the first and the third plot, speaking of gesture control, look alike but differ greatly from the second one. The second task focuses more on selecting and comparing the characteristics of a building than on the actions Zoom and Navigate. Thus, the users performed achieved different results. Section 6.2.3 takes a closer look on the struggle of some users to zoom using gesture control.

Fig. 6.2 shows a boxplot of the amount of activities the users needed to perform a task. The results are similar to Fig. 6.1, but reversed. Voice control achieves better

results in any case. This is unintuitive, as on might expect the number of activities to correlate with the amount of time spent on a task. The main reason for these results is

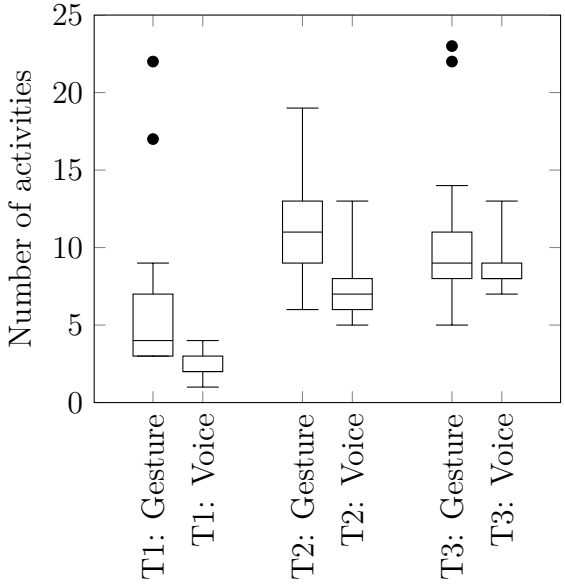



Figure 6.2.: Amount of activities performed per task.

the fact, that voice control is less prone to errors than gesture control. Given that the device recognized the user's utterance, it usually understands the correct words and hence, performs the desired action. The speech recognition sometimes fails in a sense, that it does not understand the correct words. This issue is further discussed in Section 6.2.3. Gesture control leads to errors more often. When using the Zoom action, the users perform a Two-handed Tap-and-Hold. For zooming in, they drag both the hands from the middle of the field of vision to the outer bounds. If they then want to zoom further in, they put their hands in the middle of the field of vision again and repeat the gesture. In some cases, the device does not recognize that the first gesture is already finished. Therefore, it zooms out again, as the hands move into the center again, while the Zoom gesture is not completed. The same goes for the Navigate action, where the users might drag the visualization to the left (by moving their hand to the right) and move their hand back to the left. The application continues with the initial gesture and sees, that the hand goes from the right to the left and therefore moves the application back.

Another fact that contributes to gesture control being more prone to errors is, that users, when examining an island, accidentally tap on the ocean instead of tapping on a building. The application recognizes this as a Deselect command and goes back to the main state. However, the user intended to do a Select action and select a certain region / building. The user then has to select the island again and select the region. So if the initial action was meant to be to select a package, the user performs three actions (Deselect, Select island, Select package) instead of one due to a mistake. This also adds up in the statistics shown in Fig. 6.2. This mistake does not happen in voice control due to the distinction between Select and Deselect. If a user looks on the ocean and says *Select*, IslandViz does not react, as this interaction is not intended. Instead, the user must say *Deselect* or say *Select* while focusing a region / building.

### 6.2.2. Analysing the Results of the SUS

As described in Sec. 4.6, the results of the SUS can be used to display the satisfaction of the users on a scale from 0 to 100, where 100 is the best possible result. The questionnaire made use of the SUS twice, once for gesture control and once for voice control. The results can be found in Appendix B.
Gesture control achieved an SUS score of 62.81, while voice control ended up with a score of 66.09. This means, that on average, the users were slightly more satisfied when navigating via voice control. The difference between both values is 3.28. As the range of the rating goes from 0 to 100, this value is rather low and not meaningful. In [30], Bangor et al. proposed a scale to evaluate a SUS score. This scale is presented in Fig. 6.3. Both scores for gesture and voice control are classified as OK on this scale. In terms of acceptability, gesture control places right on the line between Low Marginal and High Marginal, whereas voice control is High Marginal. Hence, it is up to one's interpretation, whether both control types bring an equal satisfaction or if the users find voice control more satisfying.

The ratings of the statement *Please indicate which type of navigation (voice control*

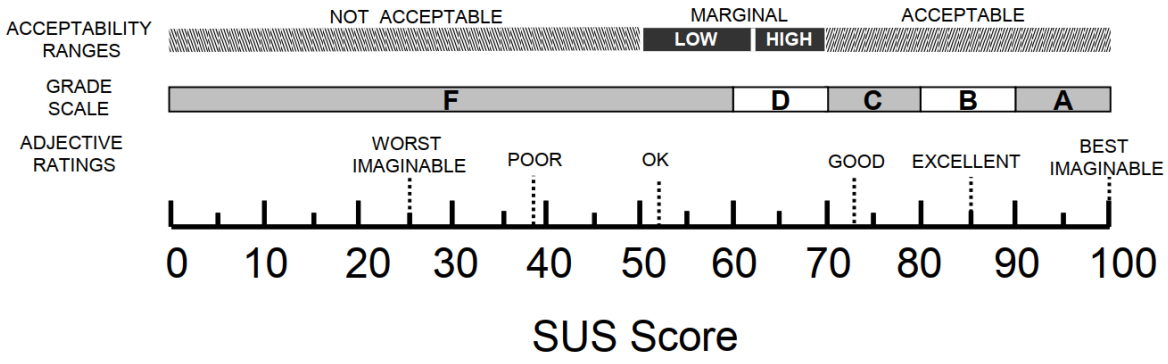



Figure 6.3.: Scale to evaluate different SUS scores.

*or gesture control) you prefer* end up similarly. Six people prefer voice control and five prefer gesture control, while the remaining five users were undecided (see Appendix C). In both cases, voice control achieved better results, although the difference is small.

The results of the ratings for the questions *How easy did you find it to solve the tasks with gesture control?* and *How easy did you find it to solve the tasks with voice control?* are, similar to the preceding two ratings, balanced. For gesture control, the results are more spread, but the overall impression remains the same. Appendix C gives an overview on the exact results.

Even though it would be exaggerated to state that the average users are clearly more satisfied using voice control, the results show, that they are **not less satisfied**. This can be seen as a success for voice control, as it could replace gesture control without lowering the overall satisfaction of the users. Additionally, voice control offers more possibilities in future development, such as searching particular islands by name. In other words, voice control is equal to gesture control, in terms of user satisfaction, but offers more possibilities overall. Thus, the results from examining the user satisfaction is a striking argument for continuing the development of voice control.

### 6.2.3. Review of the User Comments

Following the questionnaire, a text box was provided, where each user could give feedback about the user study. A lot of the users have given out criticism towards IslandViz and voice control itself. Furthermore, observing the users while performing the tasks gave a lot of information about their usage of IslandViz as well. Seeing several users performing the same tasks differently showed both advantages and disadvantages of voice control over gesture control, that were not that obvious during the development. Both the user feedback and the observations are discussed in this chapter.
These results can be categorized in two categories.

- Comparing both control types
- Criticism and suggestions for improvements for voice control

Both aspects are discussed in this section in regards of the user feedback and the observations.

#### Comparison between gesture and voice control

The majority of the users saw the addition of the speech processing in IslandViz as an improvement in terms of zooming in and out. Both the observations and the feedback show, that users, especially new and unpractised ones, struggle to perform the Two-Hand Tap-and-Hold. A lot of times the gesture was not recognized by the system, which led to frustration among some users. Furthermore, the zoom action always zooms relative to the center of the content pane. This, however, seems to be not too important to most users, as it is not once mentioned in the user feedback. But it led to the users using the drag actions more often when using gesture control than when using voice control. With voice control, they could directly zoom towards the targeted position and, therefore, did not need to make any adjustments using the drag actions. This most likely contributed to the raise of the efficiency.

As mentioned above, gesture control can be frustrating sometimes. A user stated, that *Gesture control becomes exhausting over time. Especially zooming.* This supports the observations mentioned above. Another user commented that the *gesture control is unnatural* and that the arm and fingers *cramp quickly.* No comments on this were made about voice control. In summary, speech control has advantages over gesture control in terms of zooming and seems to be less exhausting.

A disadvantage of the zoom action via voice control is, that the user can not decide the distance of how much the application zooms out. The length is a fixed value. In gesture control, the users can zoom in to the application as exactly as they want to. The same goes for dragging the application. By dragging via gesture control, users can place the visualization more precisely. When using voice commands like *go left*, the application, again, moves for a fixed distance. While executing the scenarios, user comments like *is it possible to zoom for a particular distance?* were no rarity. This seemed to be especially unsatisfying when dragging the view.
The observations regarding the actions Zoom and Navigate were summed up by a user quite well. He stated that he prefers voice control for zooming and gesture control for dragging purposes. This suggests, that voice control is not in the position to replace gesture control directly, but that they complement another.

In another comment, a user saw an advantage in gesture control over voice control, as it does not make any noise. Especially in a work environment, this is an aspect that should be taken into account. Unfortunately, making noise is inherent in voice commands, as the user needs to speak out his utterance. The fact that the speech recognition works better if the users say their sentences loud and clearly contributes to this disadvantage of voice control even more.

## Criticism and suggestions to improve voice control

One of the most frequently cited criticisms is the delay of the speech processing. As a short reminder, the language processing procedure is as follows:

1. User says an utterance
2. The Device sends the utterance to the speech recognition service and receives the sentence as a string
3. The Controller receives this string from the Device and sends it to the NLU Service and receives intent and entities.
4. IslandViz performs an action based on the received intent and entities.

Unfortunately, the most time-consuming step is the 2nd one. The service for speech processing is provided from an external company and can not be influenced by the developers of IslandViz. Therefore, the delay has to be accepted, currently. However, there are ways to make the delay less inconvenient. Both observations and user feedback came to the conclusion, that a feedback of IslandViz is necessary. This feedback can be some sort of a sound or a loading wheel that spins while the application is processing the speech input. This would be especially helpful to communicate the user whether the application is currently processing the input or whether there was no speech recognized. In some cases, users said a sentence but the system did not recognize it and, therefore, did not process it. The user, however, had no idea, whether the system understood the sentence or not. A feedback by the system would indicate if the input was recognized.
As an additional feedback the system could specify which utterance exactly is progressed. In some cases, users said sentences like *deselect* or *go right* but the system misunderstood them and interpreted the input as *this elect* or *go white*. Accordingly, the action performed by IslandViz was not what the user wanted. Especially the confusion between *deselect* and *this elect* showcases an occurrence that became frustrating

for some users. If users are exploring an island and want to deselect it, they usually use *deselect* to go back to the main state. If nothing happens for the next seconds, this can be caused by the following reasons:

- The system is still processing. Especially after restarting the NLU Service, processing the recognized speech input takes some seconds.
- The system did not recognize the utterance and therefore did not process anything.
- The system already finished processing the input. However, it misunderstood the sentence and accordingly performed an unforeseen command. In IslandViz' current state, this command might have no effect. Therefore, the user does not know, that the input was already processed.

A meaningful feedback can help the user to understand what is happening after proposing an input sentence and react to it accordingly.

Another problem which occurred several times was the confusion when using sentences like *go up* or *move left*. As described in Section 5.1.1, the visualization moves to the right when recognizing the intent `move_left`. Therefore, islands that were too far left and, consequently, were not visualized, were shifted to the right and are now displayed on the pane. This serves to simulate the view of the user going to the left. Some users were irritated by this interpretation. They expected the exact opposite, so that the visualization itself moves to the left when they say *move left*. The same confusion arose for the actions `go_up` and `go_down`. This also shows in the data gathered in the user study. In some instances, sequences like
`go_down -> go_up -> go_up`
occurred. The second command serves to revert the first command and the third command then performs the action that was originally intended by the user. Hence, the question arose, whether the present implementation of the moving actions is intuitive. Therefore, some users were asked whether these actions worked as they intended it.

The question was always asked after the users finished both the tasks and the questionnaire, so that the results are not influenced in any way. Most users found the commands intuitive and would not change it. This portrays another flaw in IslandViz, that can not be solved easily. The intuition of people works differently and a solution that satisfies both interpretations of the moving actions seems difficult. As the majority of the users was satisfied with the present solution, however, it will not be changed for the time being.

Then following remarks have also been made:

- Words or expressions like *hmmm* before the start word lead to the utterance not being recognized.
- An option to zoom back to the starting view, where every island is visible, would support the navigation.

These changes were already in mind anyway. However, the fact that the users commented about these issues as well serves as confirmation of the necessity of these changes.

## 6.3. Evaluation of the Hypotheses

In Section 3.3, three different hypotheses are put forward. Each hypothesis focuses on another aspect of usability. Based on the results described in this chapter, they can now be evaluated.
The first hypothesis discusses the system's effectiveness. The effectiveness refers to the

**H1** Adding a conversational interface to IslandViz does not affect the user's effectiveness in performing tasks.

amount of actions users need to perform certain tasks. The evaluation of the results in Section 6 shows, that this statement is not true. The addition of a conversational interface did affect the user's effectiveness, but in a positive way. The users need less activities to perform a task, hence, are more effective.

**H2** The completion of tasks in IslandViz via a conversational interface takes less time than with the use of gesture control.

Hypothesis H2 has been proven wrong as well. For each task, the users needed less time using gesture control. Hence, the addition of a conversational interface does not automatically lead to an improvement in the efficiency. The previous section discusses some of the factors that contribute to the lack of efficiency in regards to voice control and how to decrease the time spent per task. However, some factors are inevitable, currently. IslandViz makes use of an external service for speech recognition. Therefore, it sends the user's utterance to this service and receives the sentence as a string. Due to the emerging network traffic, some time, about 2-3 seconds, passes. This time frame can only be reduced by integrating and using a speech recognition service into IslandViz itself, similar to the NLU Service. It is to be weighed whether this effort is worth the resulting improvement. Other solutions, like adding a feedback that indicates that the utterance is currently being processed might also be sufficient.
Other factors that lead to the low efficiency of voice control are technical problems. The feature of speech processing is still fairly new to IslandViz and inherits bugs. For example, the connection between IslandViz and the speech recognition service aborted sometimes, if the user did not use the speech control for some time. The experience gathered during the user study shows that this problem arises roughly 2-3 minutes after the user did not use voice control, hence, the speech recognition service. This problem can be fixed by leaving the application by going to the main menu and then go back to the application again. This happens on the HoloLens itself and takes another, roughly, 5 - 10 seconds. This problem is time costly as well. First, the user must understand that the connection is aborted. If the system does not process an utterance, users might think that the application did not understand the utterance. Hence, the user

repeats saying the sentence a few times. Only if the system then still does not react, the users might understand, that the connection to the speech recognition service is aborted. They then leave and go back to the application to establish the connection anew. Usually, voice control then works again.
This is not an acceptable state for voice control to be in. The user should

- be capable of verifying, that the speech processing services are available.
- be able to restart the service in the case that the connection aborted.

The demands for speech control, of course, should be, that it is available the whole time and that the connection does not fail. However, it can not be guaranteed that each service is available any time. Especially, as an external service is used. Therefore, the above requirements on verifying and restarting the speech processing services should be considered in the further development of IslandViz to increase the efficiency of voice control.

**H3** The addition of a dialog interface affects user satisfaction.

The statement in H3 cannot be clearly answered. Both on the SUS and in the prior three questions about the user's overall satisfaction with the different control types, voice control achieved slightly better results. However, significant differences could not be observed. Based on the interpretation of the results, one might argue that the satisfaction is the same for both control types. Other might argue, that the satisfaction rose with the implementation of voice control. Both interpretations, however, mean that the overall satisfaction is not lower when using voice control than when using gestures. As discussed in 6.2.2, this can be used as a foundation to continue the development of voice control.

The original research question put forward in 1.1 is *Does a conversational interface improve the usability of IslandViz?* Due to the three aspects of efficiency, effectiveness, and satisfaction, covered by the term usability, the three hypotheses discussed in

Section 3.3 were put forward. The question oughts to be answered by the results of these hypotheses. However, the hypotheses give different answers on whether the conversational interface improves the usability. The effectiveness increased, as the users need less actions to perform a task. The efficiency, however, decreased. During the user study, more time was spent to finish tasks with voice control than with gesture control. The satisfaction can be rated as equal or slightly better. As a conclusion, the conversational interface does not improve the usability of IslandViz in its current form. Even though users need to perform less actions, they still spent more time to finish the tasks. The satisfaction has not improved significantly either. However, the usability does not aggravate either, as the efficiency improved and the satisfaction does not decrease. Hence, the usability is about equal for both gesture and voice control.

By implementing the proposed improvements for voice control, the efficiency is expected to increase and might end up being equal to the efficiency of gesture control. Furthermore, the satisfaction might increase as well, as the efficiency influences the satisfaction, as the satisfaction is likely to be influenced by whether a task takes one or 100 seconds. This, however, is only theoretical and must be examined in another study after improving voice control.

This thesis shows the main issues of voice control in IslandViz but it also shows its strength. The proposed conversational interface is not successful in improving the usability. However, it does not decrease the usability either and some issues within voice control have been discovered. Furthermore, changes have been proposed to fix these issues. With these changes, the usability should increase for voice control and, hence, for IslandViz.

# 7. Conclusion and Future Work

This thesis covers the process of creating a user study to evaluate a voice control, which was added as a feature to IslandViz, in comparison to the already existing gesture control. The study serves to answer the research question, whether a conversational interface improves the usability of IslandViz. The thesis also goes beyond the creation of the study, as additional features had to be added to IslandViz.

First, the application IslandViz was explained. This includes the purpose of the software, its architecture, used services, and the remaining implementation steps.

Next, the term usability was examined more closely. As there is no uniform definition, different definitions were considered and compared. The definition considered over the course of this work divides usability in three different aspects, effectiveness, efficiency, and satisfaction. The user study established in this thesis serves to cover all three aspects. Hence, different hypotheses were put forward to cover each aspect. These hypotheses were examined during the user study. Based on the evaluation of the different hypotheses, the original research question is to be answered.

The user study served to examine the different aspects of usability. Based on this examination, the hypotheses were evaluated. In order to set up the user study, different aspects had to be considered. Firstly, targeted audience of IslandViz was specified. The people participating in the user study ought to be part of this audience. Next, the existing features of gesture and voice control were compared. This showed, that, in terms of navigation, gesture control was superior to voice control. Therefore, new intents were added to the speech service, so that the intents provided by this service are able to cover every aspect of navigation. Next, the design of the study was dis-

cussed. The result of this comparison was, that the study is based on a within-subject design, which means, that every user completes each task both with gesture and voice control. Next, the different tasks were set up. These tasks were designed to cover each part of navigation in IslandViz. The tasks focused on evaluating the effectiveness and efficiency of the different control types. Additionally, a questionnaire was established to examine the satisfaction of the users. This questionnaire was based on the SUS.
After the concept of the user study was finished, IslandViz had to be extended by some features. While the application was already able to understand the user's intent it was not able to carry these actions out. Hence, the actions to Zoom and Navigate with speech control were implemented. The actions Select and Deselect use the same methods that are also used by gesture control. Afterwards, each intent had to be mapped to an according action. After that, the different scenarios were implemented, so that IslandViz provides an environment in which the different tasks can be performed.
Lastly, the results of the user study were examined. Based on the results, the hypotheses were evaluated and the original research question was answered.

## Conclusion

The addition of a conversational interface to IslandViz was not successful in improving its usability. The effectiveness increased, while the efficiency decreased. The satisfaction remained for the most part the same, respectively, increased slightly. Consequently, the usability remained roughly the same. However, it did not aggravate either. This can be interpreted as a good sign, as the implementation of the conversational interface still has room for improvement.
During the user study, different issues within the implementation of the conversational interface were discovered. The two biggest issues are the abortion to the external service used for speech recognition and the lack of feedback of IslandViz.
The external service itself, which translates a spoken sentence into a string, works reliably. However, the connection is often aborted after the service was not used for some minutes. Hence, the user himself has to reestablish the connection by leaving and going

back to IslandViz. After this procedure, IslandViz tries to rebuild the connection to the speech recognition service. This usually works. However, this is a cumbersome fix. Furthermore, the connection should be stable enough that this fix must only be used in exceptional cases. In the current state of IslandViz, however, this is not the case.
Another issue is, that IslandViz gives no immediate feedback to a user utterance. Hence, the users do not know, whether their input is currently processed, was already processed (but nothing happened because their input was invalid), or if it was not recognized. An immediate feedback would help the user to understand whether the speech recognition did not work as expected and why it did not work. Currently, the user proposes an utterance and waits for IslandViz to react. If IslandViz does not react, the user receives no information why there is no reaction. These information, however, are crucial for the user in order to proceed accordingly. If the system was just not able to understand his utterance, the user can repeat the sentence. If, however, the speech recognition is unavailable for some reason, the user has to react differently.
These two issues present the biggest problems inherent in the current implementation of voice control. Fixing the discovered issues in voice control is likely to increase its usability. This, however, can not be taken for granted and, thus, must be examined in a different study.

## Future Work

The user study showed that the main problem in the usability of IslandViz lies within the efficiency. Hence, the future development of the conversational interface should focus on improving this aspect. The two issues discussed above, the abortion to the external service used for speech recognition and the lack of feedback of IslandViz, should be approached first. Only if this works reliably, additional actions, such as implementing logic to the actions, should be added. This addresses the already implemented intents to count the number of components, e.g. counting the number of all compilation units within a bundle, and the intents to select the biggest or smallest component. The according intent can already be recognized by the system, however,

the action is not implemented yet. Yet, adding those actions seems unnecessary as long as the speech processing services do not work reliably. As soon as these issues are fixed, adding new functionalities is to be considered.
The feedback of the users during the user study should be taken into account as well. These changes might seem small in the overall context, however, can ensure to improve the satisfaction of the users. For example, a user suggested to implement an action to zoom back to the starting view, where every island is visible. This could support the users in navigating through IslandViz, as it gives the users a change to always return to a familiar state. This proposal and the other required changes, of course, need to be evaluated in terms of their necessity, before they implementing them.
After approaching the issues discussed in this thesis, a new user study should be executed. This study should focus mainly on the efficiency, as this aspect was worse than voice control's effectiveness and satisfaction in comparison to gesture control.
Overall, the conversational interface lives up to its promising reputation. Even though it is still in an early state of its development, it was able to keep up in terms of usability with an established control type, the gesture control. The usability did not improve due to the addition of voice control, however, it did not decrease either. Hence, further development is recommended, as voice control offers more capabilities than gesture control in general and the proposed changes seem promising in enhancing the usability of voice control as well.

# Bibliography


[1] P. Caserta and O. Zendra, "Visualization of the static aspects of software: A survey," *IEEE transactions on visualization and computer graphics*, vol. 17, no. 7, pp. 913–933, 2010.

[2] F. B. Abreu, M. Goulão, and R. Esteves, "Toward the design quality evaluation of object-oriented software systems," in *Proceedings of the 5th International Conference on Software Quality, Austin, Texas, USA*, 1995, pp. 44–57.

[3] Y. Legat, "Visualisierung osgi-basierter software architektur in augmented reality," 2018.

[4] S. Carpendale and Y. Ghanam, "A survey paper on software architecture visualization," University of Calgary, Tech. Rep., 2008.

[5] M. Misiak, "Immersive exploration of osgi based soft-ware architectures in virtual reality," 2017.

[6] A. Baranowski, P. Seipel, and A. Schreiber, "Visualizing and exploring osgi-based software architectures in augmented reality," in *Proceedings of the 24th ACM Symposium on Virtual Reality Software and Technology*. ACM, 2018, p. 62.

[7] M. Schaller, "Visualisierung von osgi-basierten softwarearchitekturen - eine vergleichende evaluation der usability in 2d und virtual reality," 2019.

[8] (2019) Gestures - mixed reality. [Online]. Available: https://docs.microsoft.com/en-us/windows/mixed-reality/gestures

[9] P. Seipel, A. Stock, S. Santhanam, A. Baranowski, N. Hochgeschwender, and A. Schreiber, "Adopting conversational interfaces for exploring osgi-based software architectures in augmented reality," in *Proceedings of the 1st International Workshop on Bots in Software Engineering*. IEEE Press, 2019, pp. 20–21.

[10] A. Stock, "Conversational interface for a visualization software," 2019.

[11] A. L. Tavares and M. T. Valente, "A gentle introduction to osgi," *ACM SIGSOFT Software Engineering Notes*, vol. 33, no. 5, p. 8, 2008.

[12] A. Schreiber and M. Misiak, "Visualizing software architectures in virtual reality with an island metaphor," in *International Conference on Virtual, Augmented and Mixed Reality*. Springer, 2018, pp. 168–182.

[13] S. Zur and A. Tröltzsch, "Optimization of the dlr space liner inside the integration environment rce," *Engineering Optimization, CRC Press*, pp. 757–761, 2014.

[14] M. Sippel, "Introducing the spaceliner vision," in *7th International Symposium on Launcher Technologies, Barcelona, Spain*, 2007.

[15] M. Grinberg, *Flask web development: developing web applications with python*. " O'Reilly Media, Inc.", 2018.

[16] H. Graham, H. Y. Yang, and R. Berrigan, "A solar system metaphor for 3d visualisation of object oriented software metrics," in *Proceedings of the 2004 Australasian symposium on Information Visualisation-Volume 35*. Australian Computer Society, Inc., 2004, pp. 53–59.

[17] C. Knight and M. Munro, "Comprehension with [in] virtual environment visualisations," in *Proceedings Seventh International Workshop on Program Comprehension*. IEEE, 1999, pp. 4–11.

[18] S. Kopp, L. Gesellensetter, N. C. Krämer, and I. Wachsmuth, "A conversational agent as museum guide–design and evaluation of a real-world application," in

*International Workshop on Intelligent Virtual Agents*. Springer, 2005, pp. 329–343.

[19] M. Lange, J. Hjalmarsson, M. Cooper, A. Ynnerman, and V. Duong, "3d visualization and 3d and voice interaction in air traffic management," in *The Annual SIGRAD Conference. Special Theme-Real-Time Simulations. Conference Proceedings from SIGRAD2003*, no. 010. Linköping University Electronic Press, 2003, pp. 17–22.

[20] S. Bieliauskas and A. Schreiber, "A conversational user interface for software visualization," in *2017 IEEE Working Conference on Software Visualization (VISSOFT)*. IEEE, 2017, pp. 139–143.

[21] J. Nielsen, *Usability engineering*. Elsevier, 1994.

[22] A. Abran, A. Khelifi, W. Suryn, and A. Seffah, "Usability meanings and interpretations in iso standards," *Software quality journal*, vol. 11, no. 4, pp. 325–338, 2003.

[23] T. Jokela, N. Iivari, J. Matero, and M. Karukka, "The standard of user-centered design and the standard definition of usability: analyzing iso 13407 against iso 9241-11," in *Proceedings of the Latin American conference on Human-computer interaction*. ACM, 2003, pp. 53–60.

[24] M. M. Punt, C. N. Stefels, C. A. Grimbergen, and J. Dankelman, "Evaluation of voice control, touch panel control and assistant control during steering of an endoscope," *Minimally Invasive Therapy & Allied Technologies*, vol. 14, no. 3, pp. 181–187, 2005.

[25] M. Whitlock, E. Harnner, J. R. Brubaker, S. Kane, and D. A. Szafir, "Interacting with distant objects in augmented reality," in *2018 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*. IEEE, 2018, pp. 41–48.

[26] J. Nielsen. (2000) Why you only need to test with 5 users. [Online]. Available: https://www.nngroup.com/articles/why-you-only-need-to-test-with-5-users/

[27] W. Hwang and G. Salvendy, "Number of people required for usability evaluation: the 10±2 rule," *Communications of the ACM*, vol. 53, no. 5, pp. 130–133, 2010.

[28] R. Alroobaea and P. J. Mayhew, "How many participants are really enough for usability studies?" in *2014 Science and Information Conference*. IEEE, 2014, pp. 48–56.

[29] J. Brooke *et al.*, "Sus-a quick and dirty usability scale," *Usability evaluation in industry*, vol. 189, no. 194, pp. 4–7, 1996.

[30] A. Bangor, P. Kortum, and J. Miller, "Determining what individual sus scores mean: Adding an adjective rating scale," *Journal of usability studies*, vol. 4, no. 3, pp. 114–123, 2009.

# A. Introduction to the User Study

## German version (original)

Liebe StudienteilnehmerIn,
vielen Dank für Ihr Interesse und Ihre Mithilfe an der Usability-Studie für meine Bachelorarbeit. In der Arbeit wird die Gebrauchstauglichkeit einer Sprachsteuerung untersucht und mit der bestehenden Gestensteuerung verglichen.
Die Studie befasst sich mit einem Programm, welches die erweiterte Realität (englisch: augmented reality, kurz AR) als Darstellungsmedium verwendet.
Augmented Reality beschreibt die Abbildung von virtuellen Objekten in einem realen Umfeld. Dies kann beispielsweise durch die Microsoft HoloLens umgesetzt werden.
Die in dieser Studie untersuchte Applikation wird dazu genutzt, Architekturen von Softwareprojekten darzustellen, welche auf OSGi basieren. Dazu werden die OSGi-Komponenten Bundles, Packages und Classes durch Inseln, Regionen und Gebäude visualisiert.
Um die Softwarearchitektur erkunden zu können, kann der Nutzer mittels Gestiken und Sprachbefehlen durch die Anwendung navigieren. Diese Studie untersucht, welche Modalitäten eine bessere Usability für den Benutzer bereitstellen.
Bitte füllen Sie als nächstes die folgenden Fragen aus. Danach erhalten Sie eine Einführung in die DLR-Software IslandViz, welche zur Ausführung der Studie benutzt wird. Im Anschluss daran werden Sie gebeten, mehrere Aufgaben zu bewältigen, um beide Navigationsarten in einem Beispielsszenario anzuwenden.

## English version

Dear study participant,

Thank you very much for your interest and your assistance in the usability study for my bachelor thesis. The thesis examines the usability of a voice control and compares it to the existing gesture control.

The study focuses on an application that uses augmented reality as a visualization tool.

Augmented reality describes the illustration of virtual objects in a real environment. This, for example, can be achieved by the Microsoft HoloLens.

The application examined in this study is used to depict architectures of software projects based on OSGi. In order to do so, the OSGi components bundles, packages, and classes are visualized by islands, regions, and buildings.

Using gestures and speech commands, the user can navigate through the application to explore the software architecture. This study examines which modality provides a better usability for the user.

Please answer the following questions. Thereafter, you will receive an introduction in the DLR software IslandViz, which is used for the purpose of executing this study. Afterwards, you will be asked to perform several tasks to apply both navigation types in an example scenario.

# B. SUS Results

| | 1 | 2 | 3 | 4 | 5 | Score |
|---|---|---|---|---|---|---|
| I think that I would like to use the gesture control in IslandViz. | 1 | 3 | 7 | 3 | 2 | 2.13 |
| I found the gesture control in IslandViz unnecessarily complex. | 7 | 2 | 3 | 3 | 1 | 2.69 |
| I thought the gesture control in IslandViz was easy to use. | 0 | 3 | 5 | 6 | 2 | 2.44 |
| I think that I would need the support of a technical person to be able to use the gesture control in IslandViz. | 8 | 4 | 1 | 1 | 2 | 2.94 |
| I found the various functions in the gesture control in IslandViz were well integrated. | 2 | 1 | 3 | 8 | 2 | 2.44 |
| I thought there was too much inconsistency in the gesture control in IslandViz. | 5 | 7 | 0 | 2 | 2 | 2.69 |
| I would imagine that most people would learn to use the gesture control in IslandViz very quickly. | 0 | 3 | 3 | 6 | 4 | 2.69 |
| I found the gesture control in IslandViz very cumbersome to use. | 3 | 3 | 4 | 5 | 1 | 2.13 |
| I felt very confident using the gesture control in IslandViz. | 2 | 3 | 2 | 6 | 3 | 2.31 |
| I needed to learn a lot of things before I could get going with the gesture control in IslandViz. | 6 | 5 | 1 | 2 | 2 | 2.69 |
| Summed up score: | | | | | | 62.81 |

Table B.1.: Summed up SUS scores for gesture control.

| | 1 | 2 | 3 | 4 | 5 | Score |
|---|---|---|---|---|---|---|
| I think that I would like to use the voice control in IslandViz. | 2 | 1 | 3 | 7 | 3 | 2.5 |
| I found the voice control in IslandViz unnecessarily complex. | 5 | 6 | 4 | 1 | 0 | 2.94 |
| I thought the voice control in IslandViz was easy to use. | 0 | 3 | 1 | 10 | 2 | 2.69 |
| I think that I would need the support of a technical person to be able to use the voice control in IslandViz. | 5 | 3 | 5 | 3 | 0 | 2.63 |
| I found the various functions in the voice control in IslandViz were well integrated. | 1 | 4 | 3 | 6 | 2 | 2.25 |
| I thought there was too much inconsistency in the voice control in IslandViz. | 6 | 5 | 4 | 0 | 1 | 2.94 |
| I would imagine that most people would learn to use the voice control in IslandViz very quickly. | 1 | 1 | 4 | 7 | 3 | 2.63 |
| I found the voice control in IslandViz very cumbersome to use. | 5 | 4 | 5 | 1 | 1 | 2.69 |
| I felt very confident using the voice control in IslandViz. | 1 | 2 | 4 | 6 | 3 | 2.5 |
| I needed to learn a lot of things before I could get going with the voice control in IslandViz. | 5 | 6 | 2 | 1 | 2 | 2.69 |
| Summed up score: | | | | | | 66.09 |

Table B.2.: Summed up SUS scores for voice control.

# C. General Questions

| Please select your gender. | | |
|---|---|---|
| Answer | Number | Percentage |
| Masculin | 10 | 62.5% |
| Feminine | 6 | 37.5% |
| Divers | 0 | 0% |

| I have already worked with OSGi. | | |
|---|---|---|
| Answer | Number | Percentage |
| Yes | 9 | 56.3% |
| No | 7 | 43.8% |

| I have already worked with RCE. | | |
|---|---|---|
| Answer | Number | Percentage |
| Yes | 7 | 43.8% |
| No | 9 | 56.3% |

| I have used the HoloLens before. | | |
|---|---|---|
| Answer | Number | Percentage |
| Yes | 11 | 68.8% |
| No | 5 | 32.3% |

| I have already used the implementation of IslandViz on HTC Vive. | | |
|---|---|---|
| Answer | Number | Percentage |
| Yes | 9 | 56.3% |
| No | 7 | 43.8% |

| Please indicate which type of navigation (voice control or gesture control) you prefer. | | |
|---|---|---|
| Answer | Number | Percentage |
| Voice control | 6 | 37.5% |
| Gesture control | 5 | 31.3% |
| Undecided | 5 | 31.3% |

| How easy did you find it to solve the tasks with gesture control? | | | How easy did you find it to solve the tasks with voice control? | | |
|---|---|---|---|---|---|
| Answer | Number | Percentage | Answer | Number | Percentage |
| 1 - Very easy | 3 | 18.8% | 1 - Very easy | 1 | 6.3% |
| 2 | 7 | 43.8% | 2 | 8 | 50% |
| 3 | 4 | 25% | 3 | 7 | 43.8% |
| 4 | 2 | 12.5% | 4 | 0 | 0% |
| 5 - Very difficult | 0 | 0 | 5 - Very difficult | 0 | 0 |